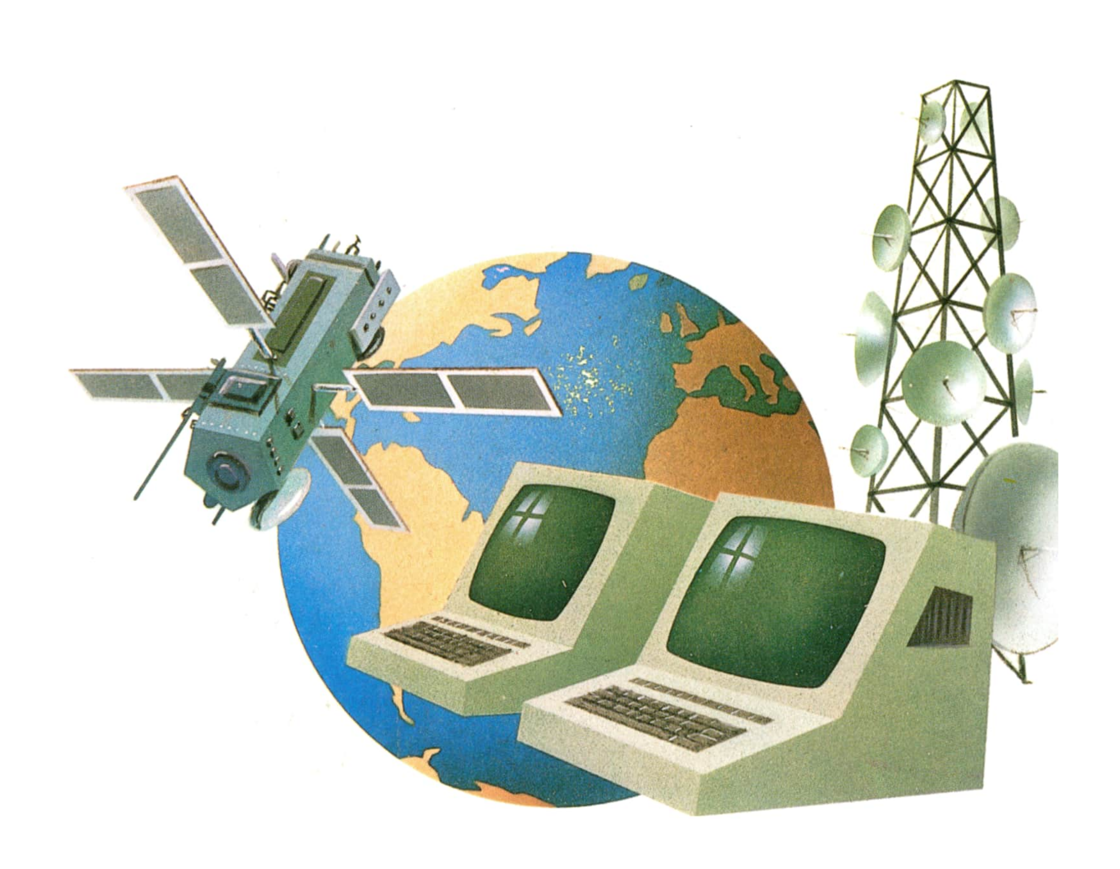
# APRENDE INFORMATICA



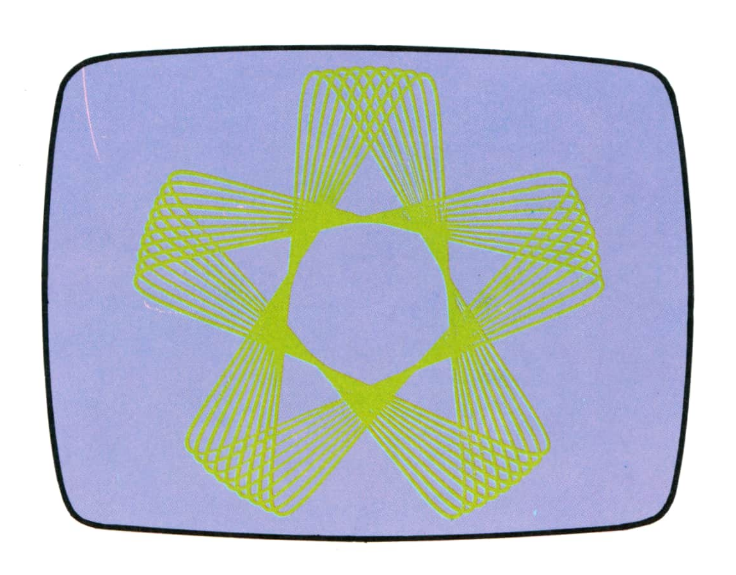
océano





# aprende informática

# aprende informática



océano

**Es una obra del**

GRUPO EDITORIAL OCEANO

**Presidente**
José Lluis Monreal

**Director General**
José M.ª Martí

**Director General de Publicaciones**
Carlos Gispert

EQUIPO EDITORIAL

**Dirección**
Carlos Gispert

**Subdirección**
José Gay

**Edición**
Sebastià Puigserver

**Diseño**
Ferràn Javaloyes

**Maquetación**
Ferràn Javaloyes
Esther Amigó

**Ilustración**
Jordi Jaumandreu
Juan Carlos Martínez Tajadura

**Dirección de la obra y redacción**
Xavier Serra
*Informático y profesor de Informática en la Universidad Autónoma de Barcelona*

**Dirección Técnica**
Mercè Feliu

**Secretaria Técnica**
Esther Amigó

**Dirección de Producción**
José Gay

**Equipo Producción**
Antonia Pérez
Antonio Surís
Ione Beobide
Alex Llimona

A mis hijas: Vera y Circe
X. Serra


Paseo de Gracia, 24-26
Teléfonos: 317 45 08* y 301 01 82*
Télex: 51.735 exit e
08007 Barcelona (España)



Impreso en España - Printed in Spain

ISBN: 84-7069-584-3
Dep. Legal B-1276-88 (Ab)
Impresión: GRAFOS, S.A.

# INTRODUCCIÓN

Un ordenador o computadora es una máquina que se acuerda de todo. Tener un ordenador es como ser el dueño de un vídeojuego sencillo pero dotado de una memoria prodigiosa y que, además, lo hace todo muy deprisa.



*Cada día más las computadoras están presentes en todas las actividades de nuestra vida. Estas máquinas, intrigantes pero de gran utilidad, no sólo pueden hacerte los deberes como si tuvieras a tu entera disposición el robot de la derecha, sino que sirven también para comunicarse, mediante terminales y satélites, con todo el mundo a la velocidad de la luz.*



## ¿Me puede hacer los deberes?

Ya te gustaría, ¿verdad? Pues sí, te los puede hacer. Pero primero tienes que decirle cómo se hacen todos los deberes.

## ¿Y cómo se lo digo?

En su idioma, en el idioma de las máquinas.

## ¿Las máquinas hablan?

Claro que hablan. Tampoco es tan difícil. Hablar sólo es conseguir entenderse con los demás. Esto las personas lo hacemos diciendo lo que pensamos y entendiendo lo que oímos. Es decir, utilizando un lenguaje hablado. Pero también nos podemos entender leyendo y escribiendo. Esto es el lenguaje escrito. También existe el lenguaje de los signos que es bastante universal.

Pues las máquinas «hablan» porque comprenden lo que les decimos y nosotros entendemos lo que nos contestan. Lo que pasa es que utilizan (hablan), otro idioma. Lo que tampoco es tan raro. Los ingleses hablan inglés y los chinos, chino. Y, si quieres hablar con ellos, debes aprender su idioma o ellos el tuyo o entenderos por signos.
De momento las máquinas no entienden nuestro idioma (aunque se está tratando de que lo aprendan; de momento ya hay llaveros que suenan cuando les silbas) y debemos resignarnos a aprender el suyo. No es difícil. Para decirle a una máquina lo que queremos que haga, sólo hay que apretar botones y, para entender lo que te dice, debes escuchar y además saber mirar y leer.

Veamos dos ejemplos significativos.

*La bombilla y el ascensor*
Si quieres encender una luz, mueves un interruptor y la bombilla se enciende. Es como si le dijeras a la luz: «¡enciéndete!» y ella te entendiera y te obedeciera. Si no lo hace, esto sólo indica que la bombilla está fundida, o que el interruptor no funciona, o que el cable está roto, o que no hay electricidad. En definitiva, es como si le hablaras a una pared.
Cuando tomas un ascensor, le dices «AL CUARTO», y la máquina te obedece. Para indicarle que debe ir al piso cuarto tienes que cerrar las puertas y apretar el botón que pone -4-. El ascensor puede no hacerte caso, pero ya se sabe que los ascensores se estropean a menudo.
Las máquinas, además de escuchar y comprender, nos hablan y esperan que les obedezcamos. Piensa en los semáforos: encienden su luz roja y todos los conductores paran sus coches, y esperan obedientes
(e impacientes) a que se apague la luz roja y se encienda la verde, mientras los peatones se apresuran a cruzar, deseosos de que la luz no cambie.
*¿Y cómo sabe el semáforo que tiene que cambiar y el ascensor que tiene que ir al cuarto?* Vamos a explicártelo. Y también cómo puedes hacer que una computadora te ayude en tus tareas escolares.



*Aunque estamos tan acostumbrados a pulsar el botón del ascensor para dirigirnos al piso de casa, lo cierto es que el ascensor es una compleja máquina, que necesita una memoria para cumplir nuestra orden.*

## LA BOMBILLA Y EL ASCENSOR

### Las máquinas

Hemos empezado diciendo que un ordenador o computadora era una máquina capaz de hablarnos y que, para entenderla, debes aprender «su idioma». Así que empezaremos dejando claro que una máquina es un aparato compuesto generalmente de muchas piezas que hace un trabajo a cambio de recibir energía. Todo el mundo está de acuerdo en que la primera máquina utilizada por el hombre fue el palo, la piedra o el hueso. Si te parece raro llamarle máquina a un palo, piensa en las palancas. Tienes que levantar una piedra que pesa mucho. Le pones un palo debajo, colocas un punto de apoyo, aprietas y la piedra se levanta. El palo amplifica tu fuerza, luego es una máquina: realiza un trabajo si le das energía.

*Energía muscular*
Todo el mundo está de acuerdo también en que la primera energía que se empleó fue la muscular. Primero se utilizaban estas máquinas primitivas tal como se encontraban. Pero enseguida el hombre descubrió que, si afilaba una piedra, cortaba mejor y que un palo con punta pinchaba más que otro que no la tuviera. Este descubrimiento representó un invento extraordinario. Al hecho de que el hombre sea capaz de cambiar las cosas a su conveniencia se debe el que la civilización sea lo que es. Mucha gente piensa que esta capacidad es lo que diferencia al hombre de los animales, pero se olvidan de los panales de las abejas, de las presas de los castores, de los nidos de los pájaros y de los hormigueros. Los hombres se acostumbraron a usar herramientas.
Y no se contentaron con las que tenían. Inventaron más. Con un palo y una cuerda hicieron un arco. Para fabricar la flecha reunieron madera, piedra y plumas.

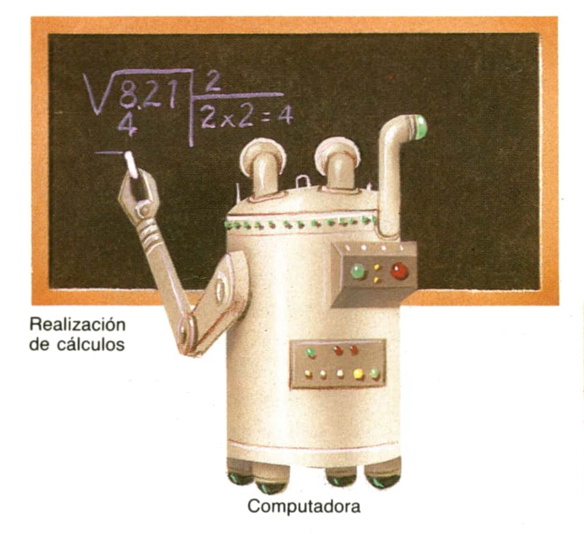
## Energía y trabajo

El libro de física dice que la energía es la capacidad de hacer un trabajo. Que el trabajo es una fuerza por un espacio y que una fuerza es una masa por una aceleración. Dicho así parece muy complicado. Pero es que, aunque parezca castellano, este trozo está escrito en «físico». Traducido al castellano normal quiere decir que la función del trabajo es cambiar las cosas. De sitio o de forma.
Con el invento del arco, los hombres aprendieron a usar la energía. Concentraban toda la energía necesaria en curvar un palo para lanzar otro de punta afilada. Ésta es una forma más complicada de aprovechar la energía que mover una piedra para cascar una nuez.
Pero todavía la energía era muscular. Más tarde, los hombres descubrieron que había muchas otras formas de energía. Y aprendieron a utilizarlas. Hicieron fuego, barcos de vela y molinos de agua. Usaban la fuerza del agua y del viento al moverse.

## Energía interna

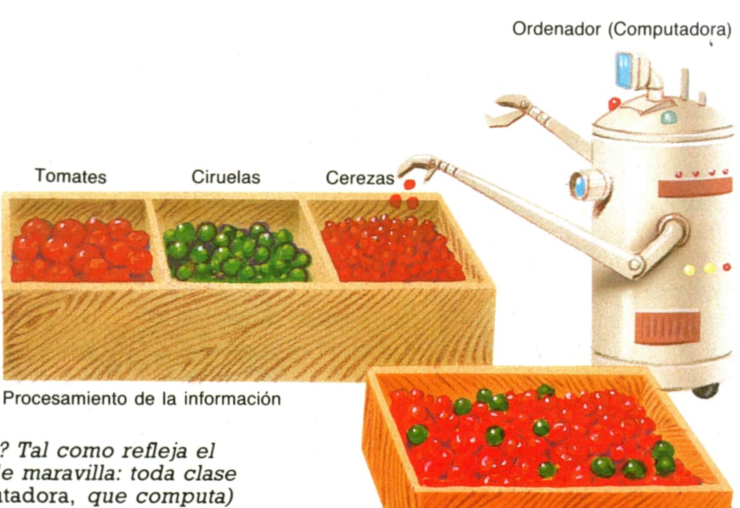
¿Por qué unas cosas queman más fácilmente que otras? ¿Qué son las llamas? Estas cosas te las explicarán con más detalle en el libro de física. Aquí sólo te diremos que todas las cosas están hechas de otras mucho más pequeñas; tan pequeñas que no las ha visto nadie. Y a estas cosas se las llama átomos.
Aunque no se les haya visto nunca, se conoce bastante bien el comportamiento

### Cálculo y procesamiento de la información



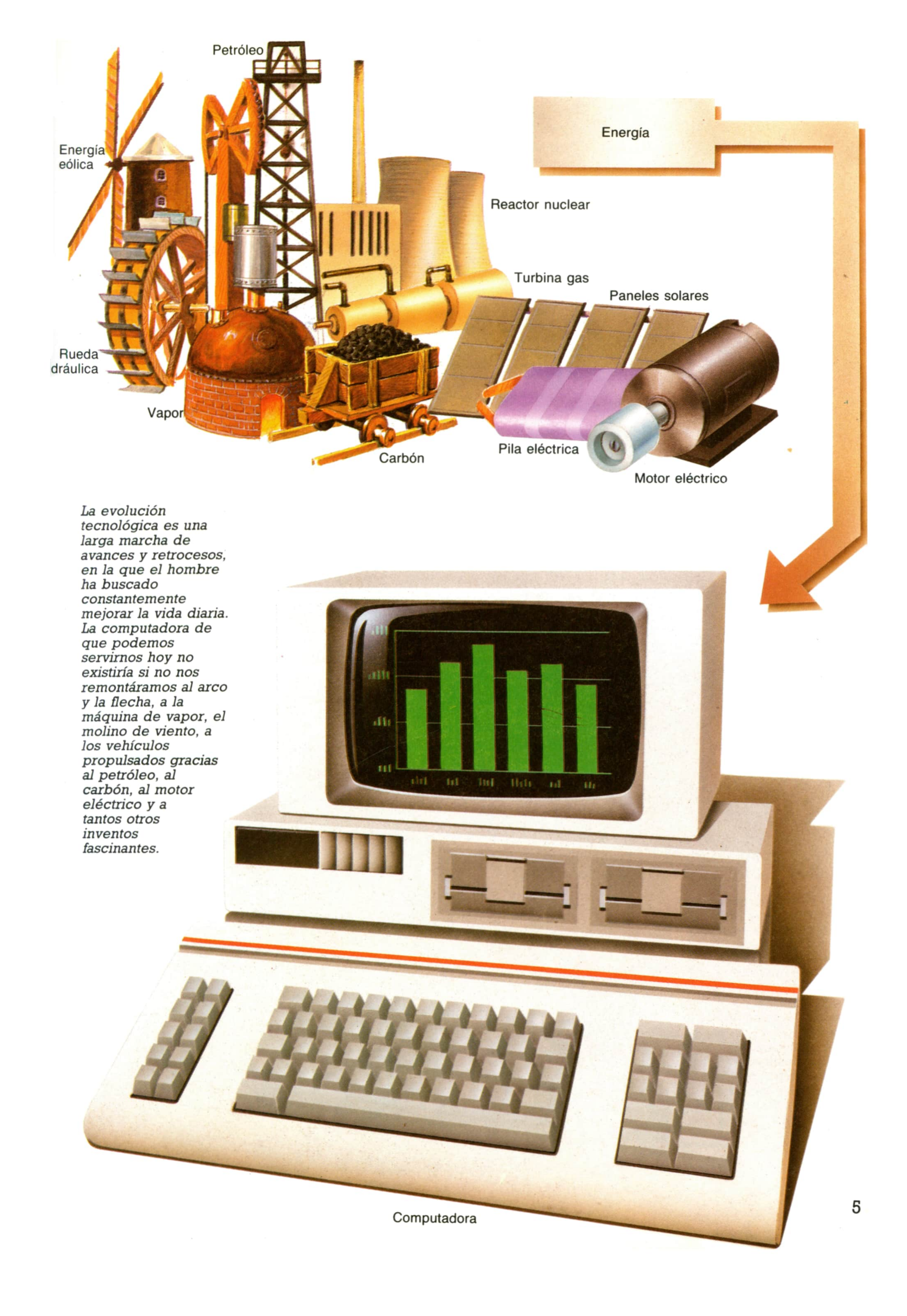
Realización de cálculos

Computadora

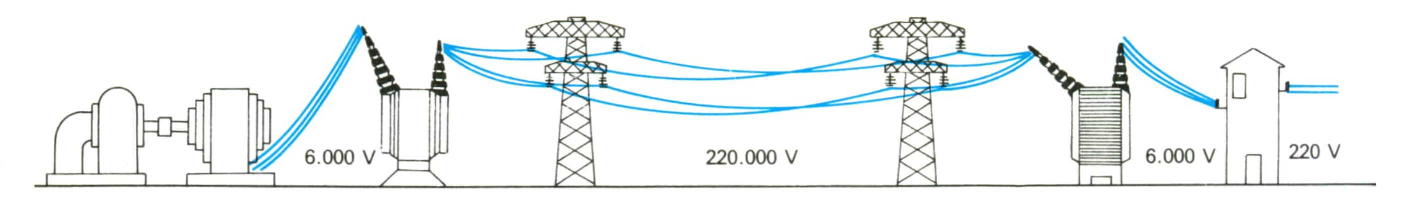


Procesamiento de la información

*¿Para qué puede servirnos una computadora? Tal como refleja el dibujo, la computadora realiza dos trabajos de maravilla: toda clase de cálculos (razón por la que se llama* computadora, *que computa) y distribuye perfectamente todo aquello que queremos saber (por ello se le llama también* ordenador, *porque ordena las cosas), naturalmente si se lo exponemos en el idioma de las máquinas.*




*La evolución tecnológica es una larga marcha de avances y retrocesos, en la que el hombre ha buscado constantemente mejorar la vida diaria. La computadora de que podemos servirnos hoy no existiría si no nos remontáramos al arco y la flecha, a la máquina de vapor, el molino de viento, a los vehículos propulsados gracias al petróleo, al carbón, al motor eléctrico y a tantos otros inventos fascinantes.*




*La electricidad ha de circular de un lugar a otro para dar energía. Para ello están los cables y las torres que cruzan nuestros caminos y campos. El dibujo superior muestra el recorrido de la corriente eléctrica desde las turbinas hasta el transformador de alta tensión, esta caseta siniestra con una calavera, que pone «NO TOCAR, PELIGRO DE MUERTE».*

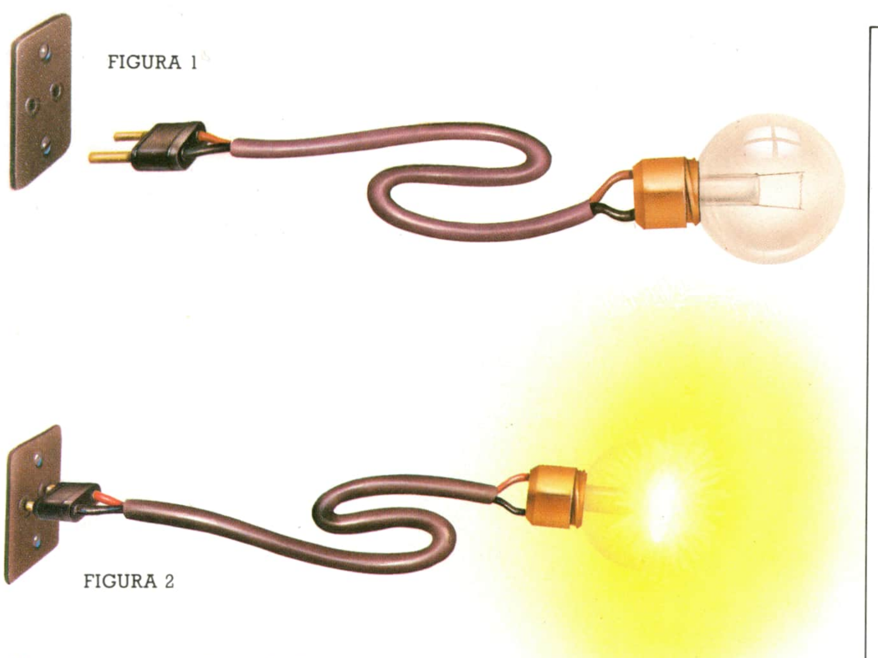
de los átomos. El fuego es el resultado de lo que pasa cuando alguna clase de átomos se mezcla con los átomos del oxígeno del aire. De esta combinación sale calor. Es pues una energía que está dentro de los cuerpos (en unos más y en otros menos). En algunos cuerpos, la energía interna sale fácilmente, como en el caso de la paja, con una cerilla, por ejemplo, y, en otros, de ninguna manera, como en el caso de una piedra.

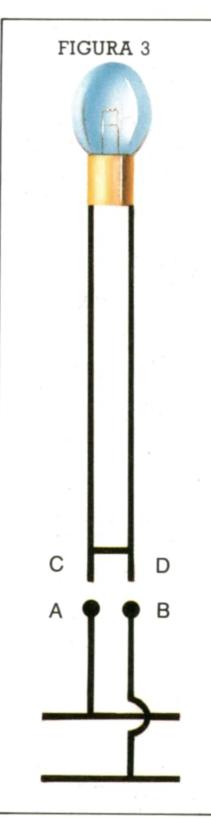
## Energía eléctrica

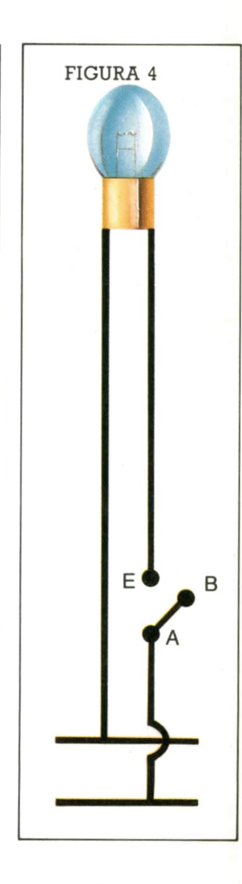
Como el fuego, la electricidad es una forma de energía que está en los átomos. Es también la más fácil de usar porque es la más sencilla de transportar. Has visto los cables de alta tensión y los que hay en tu casa. Las tuberías de agua son mucho más anchas, pero circula por ellas muchísima menos energía. La energía eléctrica sólo tiene una desventaja y es que no se puede almacenar convenientemente. Las pilas se acaban muy rápidamente.
La electricidad, como el agua, necesita moverse para dar energía. Y para moverse, la electricidad necesita tener una entrada y una salida. También como el agua. Te puedes imaginar la corriente eléctrica como una corriente de agua. Pero lo importante es entender que la electricidad circula, de aquí la palabra «circuito» que aparece siempre que se habla de electricidad. Un circuito es un camino cerrado. Piensa en los circuitos de las carreras de motocicletas.

## La bombilla

Una bombilla es una máquina muy sencilla. Su trabajo es dar luz. La energía que recibe es la electricidad. Para que la bombilla haga su trabajo sólo hace falta conectarla a la red. Como has visto, cada vez que enchufas algún aparato, introduces dos patillas o clavijas en dos agujeros, y esto es porque ¡recuerda! la electricidad sólo funciona si puede circular. Con una clavija le das la entrada y con la otra la salida. Mira las figuras **2** y **3**. Si el enchufe no está conectado, la bombilla no recibe electricidad y permanece apagada. Si está enchufada, se establece el circuito y la bombilla da luz.
Pero es muy complicado entrar a oscuras



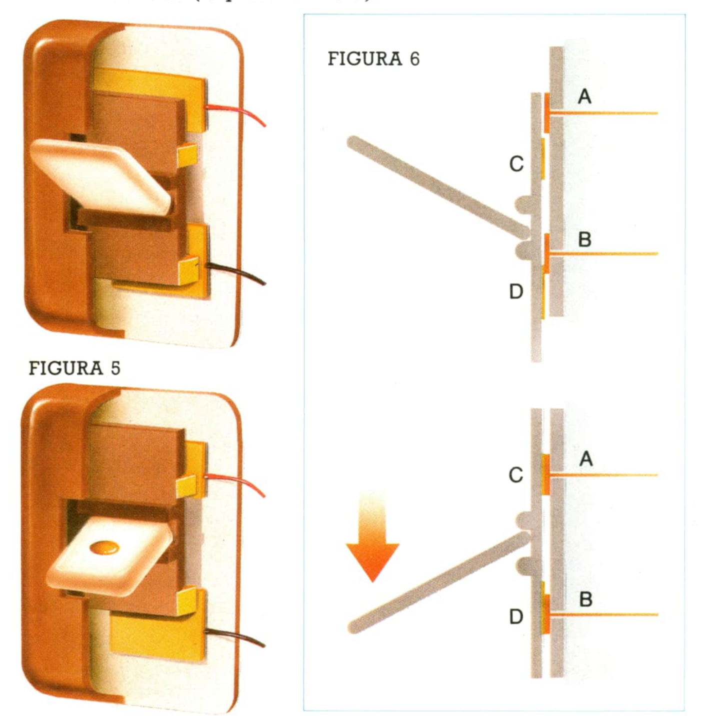

*Mientras no enchufamos la bombilla, la habitación permanece a oscuras (figura 1). Cuando hacemos la conexión, se enciende la luz (figura 2). Al encender, lo que hemos hecho es conectar las clavijas C y D de la figura 3 en los huecos A y B de la base del enchufe. La figura 4 es el esquema de un interruptor, en el que la palanca AB es la conexión permanente de una de las dos clavijas C y D de la figura 3 en el hueco del aplique B. Para encender o apagar la luz, bastará con accionar la palanca que abre o cierra el circuito.*

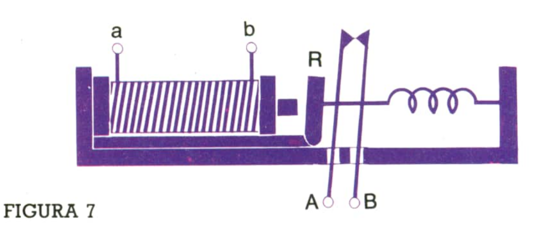
en una habitación, buscar el enchufe de la lámpara y meterlo en el aplique de la pared. Y, además, siempre da un poco de miedo manipular a oscuras con la electricidad. Así que se inventó una cosa más sencilla: el interruptor.

## El interruptor

Como para establecer el circuito hacen falta dos clavijas o patillas, podemos dejar una siempre conectada y sólo tendremos que mover la otra. Observa la figura **4**. Accionando la palanca AB, la luz se enciende y se apaga. Si tienes curiosidad por ver cómo funciona de verdad sólo tienes que mirar uno. La figura **5** y la figura **6** muestran el funcionamiento de un interruptor. Fíjate que, al mover la palanca hacia abajo, se conecta el circuito y al moverla hacia arriba se abre.
El interruptor, como su nombre indica, sirve para interrumpir un circuito eléctrico. Pero como el interruptor es una parte activa del circuito, es decir, que la corriente circula por él, debe ser tan grande como el resto de la línea. Esto suponía construir interruptores enormes.

*Figura 5, las dos posiciones del interruptor. Figura 6, funcionamiento. La clavija D se ha conectado permanentemente en el hueco B del aplique. Moviendo la palanca hacia abajo, se conecta el circuito (C penetra en A).*



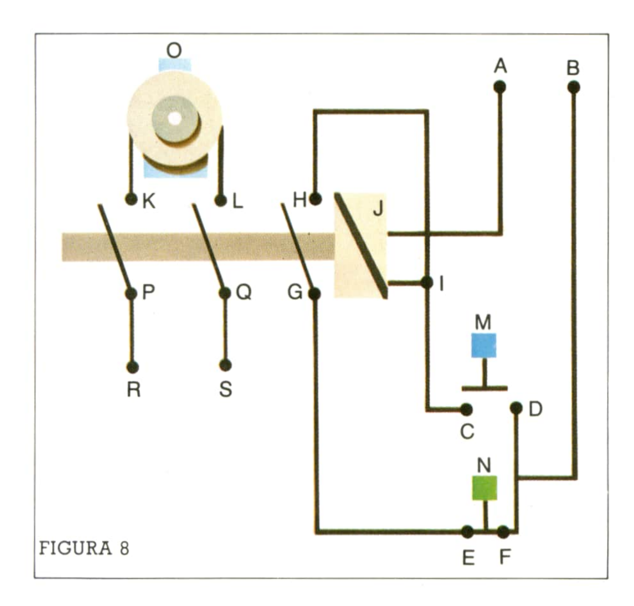

*Esquema de un relé. El interruptor AB le da la corriente al relé R y su núcleo de hierro se mueve, conectando a-b.*

Para solucionar este problema se aprovechó otras de las propiedades de los cuerpos. Aquella por la cual, si hacemos circular electricidad alrededor de un trozo de hierro, éste se convierte en un imán y, como tal, atrae a los objetos metálicos. Si aprovechamos la fuerza del imán para cerrar un contacto, podemos conectar un interruptor enorme a partir de uno más pequeño.

## El relé

Fíjate en la figura **7,** este aparato se llama relé. El interruptor AB le da corriente al relé R y su núcleo de hierro se mueve y conecta un interruptor mayor a-b. De esta manera puedes mover palancas a distancia. El principio de lo que ahora es tan normal, eso de apretar un botón y que se ponga en marcha una central eléctrica a muchos kilómetros de distancia, empezó con este modesto relé. Que utiliza, además de la eléctrica, otra forma de energía: la magnética.
Para llegar a cambiar la palanca por el botón, mucho más fácil de manejar, se diseñó el esquema de la figura **8**. Si lo miras verás que es mucho más complicado que los anteriores. Debes aprender a leer los esquemas como sabes leer un libro. De hecho es lo mismo. Son dibujos que podemos comprender. Cierto que los esquemas se parecen más a los jeroglíficos que al alfabeto, pero es más fácil dibujar un rectángulo cruzado por una línea que escribir «bobina».

*Interpretación de un esquema*
A y B representan la entrada y salida de la corriente que alimenta al relé. J e I son las entradas de corriente de la bobina del relé. R y S son las entradas de la corriente que alimentan al motor MO. Como C y D no están conectados, la corriente no puede circular. Apretamos el botón M. C y




*Esquema de un relé para poner en marcha un motor.*

D se conectan. Se cierra el circuito y se conectan G y H, S y L y R y K. El motor se pone en marcha. Soltamos el botón y el relé no se desconecta porque la corriente sigue circulando de A a J y de I a H-G-E-F-B.
Si apretamos el botón N, se desconecta el tramo E-F, se abren los contactos H-G, S-L, R-K, y se para el motor.
De esta manera, en vez de mover un interruptor, aprietas un botón para ponerlo en marcha y aprietas otro para parar.

## El biestable

Es el siguiente paso. En vez de dos botones basta con uno solo que cambia lo que se estaba haciendo cada vez que se pulsa. Seguro que has visto alguna lámpara de mesa en la que aprietas el botón y se enciende y lo vuelves a apretar y se apaga. El esquema del biestable es más complicado pero la idea es la misma. En lenguaje «informático» diremos que es un aparato que «cambia de estado» cada vez que recibe un impulso. Cuando se inventó, no se sabía que sería la base misma de la informática.

## El ascensor

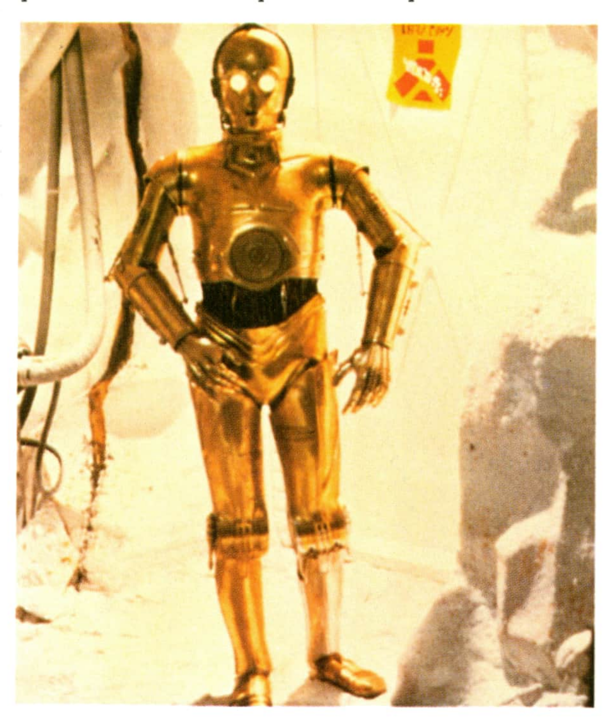
Si queremos fabricar un ascensor nos hará falta un motor, que es el encargado de subir y bajar la caja donde nos metemos, y un circuito eléctrico que haga posible todas las operaciones.

En la figura **8**, has visto el esquema de un relé que pone en marcha un motor. Hay un botón que lo pone en marcha y otro que lo para. En un ascensor el botón que lo pone en marcha es el que se aprieta. El que lo para es un brazo que se mueve cuando el ascensor pasa por allí. Si te fijas bien en los ascensores que conoces, lo verás. Acostumbra a estar en la parte de arriba, normalmente en el lado por donde se abre la puerta.
Si el ascensor sólo tiene una parada, el esquema es sencillo. El botón lo pone en marcha y el motor funciona hasta que la caja acciona la palanca, que se llama de «final de carrera».

*Un robot primitivo*

Si hay más de un piso, lo que tiene que hacerse es diseñar tantos circuitos como pisos. Cada circuito se enciende con un botón y se corta con una palanca situada al final de su recorrido previsto.
El ascensor, ese aparato que los porteros no te dejan utilizar si no tienes catorce años, es el primer robot que existió. No se parece ni al *Erredós Dedós* ni al *Cetres Peó* de la película *«La Guerra de las*

*A pesar de que cuando hablamos de un robot, nos imaginamos algo así como el R2D2* (Erredós Dedós) *de la película* La Guerra de las Galaxias, *lo cierto es que el ascensor fue el primer robot que existió.*



*Galaxias*, pero al menos sabe si tiene que subir o bajar cuando le pides que te lleve al cuarto. Si le dices al tercero, sube, y si estás en el quinto baja. Lo difícil para que un ascensor suba o baje no está en que lo haga, sino en que sepa cuándo debe hacerlo. Que suba o baje sólo depende de que el motor gire en una dirección o en la contraria. Y esto se consigue cambiando las conexiones internas del motor.

## Los semáforos

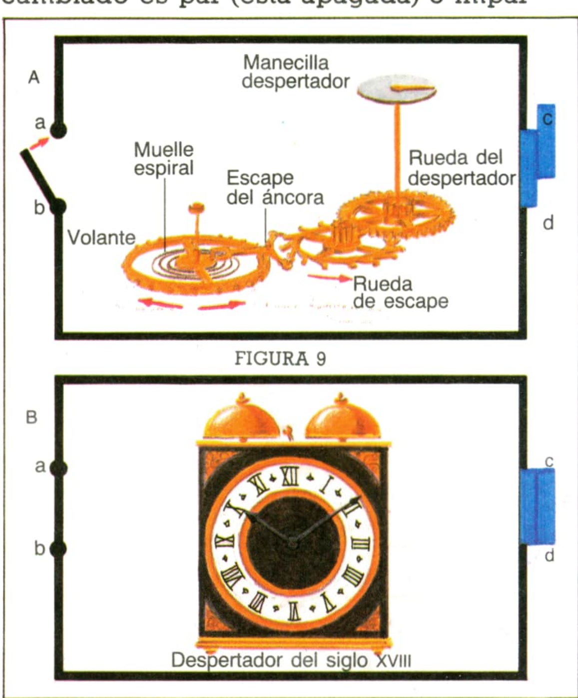
Si quieres que una luz se encienda y se apague, sólo tienes que mover un interruptor. Si quieres que lo haga de manera automática, es decir ella sola, tienes que complicar el sistema. Como cada vez que se quiere que una máquina haga algo más complicado.

Ahora quieres que se encienda y se apague cada cierto tiempo. Te hace falta un aparato que mida el tiempo: un reloj o temporizador. Y un reloj que te avisa se llama despertador. Recuerda como lo preparas. Sitúas la manecilla del despertador a la hora que quieres que suene, colocas el botón en la posición de «alarma» y te olvidas. Cuando la aguja llegue a la hora señalada, moverá la palanca que cierra el contacto y el despertador se pondrá a trabajar, sonando hasta que lo desactives, es decir, poniéndolo de nuevo en posición de reposo. La figura **9** muestra el esquema de funcionamiento de un despertador.

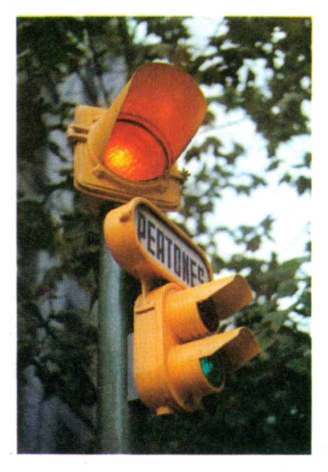
Un semáforo es un curioso aparato que posee tres luces que se encienden y se apagan cada cierto tiempo.

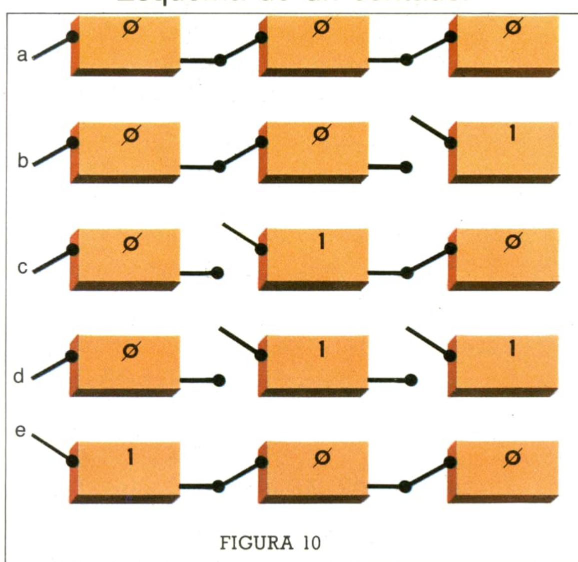
## Los contadores

Cuando explicábamos en qué consistía un biestable, decíamos que era un aparato que cambiaba de estado a cada impulso que recibía. Si conectamos un biestable a una bombilla, ésta se encenderá y, al primer impulso, se enciende, al segundo se apaga, al tercero se enciende y así sucesivamente. Podremos saber, mirando la bombilla, si el número de veces que ha cambiado es par (está apagada) o impar




*Esquema de funcionamiento de un despertador. En A, el despertador está en la posición de alarma. Así cuando la aguja llega a la hora fatal de levantarse, mueve la palanca a-b que cierra el contacto mediante la conexión de c con d, que desencadena el odioso timbre. En B, el despertador está sonando sin parar. El despertador es una maravilla de la relojería, tanto su mecanismo interno (arriba), como la magnífica figura de despertador del siglo XVIII (abajo). La tensión del muelle espiral o cuerda mueve las manecillas a través del sistema de engranajes sucesivos de las ruedas de escape y de despertador, que a su vez están conectadas con la segundera, minutera y manecilla horarias. Abajo, semáforo urbano e instalación para su funcionamiento automático. El semáforo aventaja al biestable en que puede obtener el encendido/apagado de tres luces cada cierto tiempo.*






(está encendida). Hemos inventado una máquina que sabe contar hasta dos. Si ponemos tres biestables de la manera que indica la figura **10**, verás que, al principio, (en a) las tres bombillas están apagadas. Al primer impulso (b) se enciende la primera. Al segundo (c) la segunda, al tercero (d) están encendidas las dos. Al cuarto (e) se vuelve a empezar. Mirando las bombillas podremos saber si estamos en el primer impulso, en el segundo, en el tercero o en el cuarto. Tenemos ya un aparato que sabe contar hasta siete.

Si un despertador va conectando y desconectando, este contador irá al mismo tiempo contando y, si la primera vez enciende la luz verde, la segunda lo hará la de color ámbar y la tercera la roja y luego vuelve a empezar; tenemos un semáforo.

*Arriba, esquema de funcionamiento de un contador. En **a**, las tres bombillas están apagadas. Al primer impulso **(b)**, se enciende la primera. Cuando efectuamos el segundo impulso, o sea **c**, se enciende la segunda; mientras que, al realizar el tercero **(d)**, se encenderán las dos últimas. Finalmente, al cuarto impulso **(e)**, habremos empezado de nuevo. Si a un despertador que conecta y desconecta, le aplicamos este contador, obtendremos las tres luces (verde, ámbar y rojo) del semáforo. Abajo, centro de dirección del tráfico rodado en Japón, que sincroniza todos los semáforos de Tokio.*

Esquema del motor calórico de Ericsson

FIGURA 11

*Arriba, el motor calórico de Ericsson. Cuando el eje gira, las bolas se van levantando. Si se acelera demasiado, las bolas suben más y accionan la palanca que deja escapar la presión y la velocidad disminuye. Abajo, grabado de J. Watt.*



## Los automatismos

Con las primitivas máquinas no había necesidad de automatizar los procesos, porque en realidad no había procesos. Pero, cuando Watt puso a punto la máquina de vapor, la conmoción industrial fue enorme. ¡Eso sí que es una máquina! Ya sabes lo que es. Aprovechar la fuerza del vapor del agua. Pero también era un poco peligrosa. Si se daba demasiada presión a la caldera, podía reventar. Así que había que controlarla continuamente. Para evitar este trabajo, aburrido y delicado al mismo tiempo, se diseñó el primer automatismo de la historia moderna: la *máquina de bolas de Watt,* basada en la fuerza centrífuga. La figura **11** corresponde al *motor calórico de Ericsson,* una máquina mucho más perfeccionada que la de Watt, pero que obedece en realidad a los mismos principios que los que hicieron famosa a su ilustre antepasada.

# LOS NÚMEROS

Un número es un símbolo que indica una cantidad. Mucho antes de que el hombre inventara la escritura, sintió la necesidad de registrar cantidades de una u otra forma. Por ejemplo, precisaba cuantificar el número de animales que le pertenecían. Y, como es fácil comprender, decidió utilizar con tal objeto sus propias extremidades, en especial los 10 dedos, de la mano. Así surgieron los números.

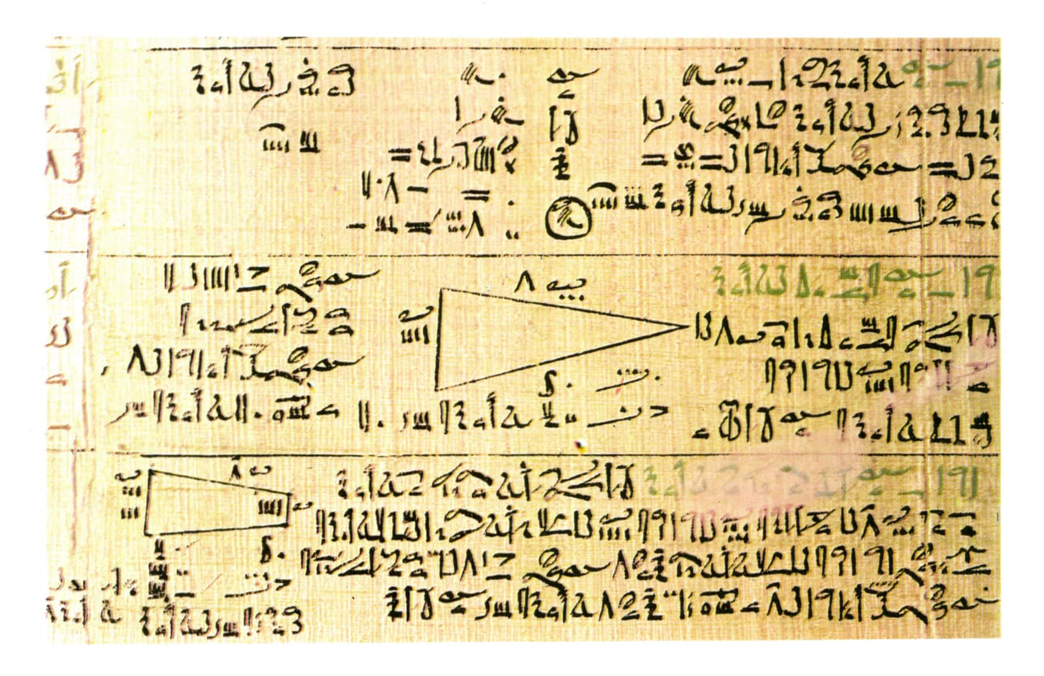
Todo el mundo está de acuerdo en que, para contar pocas cosas, los primitivos relacionaban cada cosa con un dedo. Imagínate un pastor. Mientras tiene menos de once corderos los hace pasar por delante de él y por cada uno que pasa levanta un dedo. Si no le ha nacido ninguno, ni se lo han robado, ni se lo ha comido, cada vez tiene que levantar los mismos dedos que la vez anterior. Supongamos que le nacen más corderos de los que se come o le roban. Cuando tenga más corderos que dedos nuestro pastor necesitará acordarse del número de manos que ha utilizado al contarlos. Así nacieron los sistemas de numeración. Hay casi tantos como pueblos. Y como todos pasan de una manera u otra por el diez, se puede pensar que todos empezaron contando con los dedos.

### Números escritos

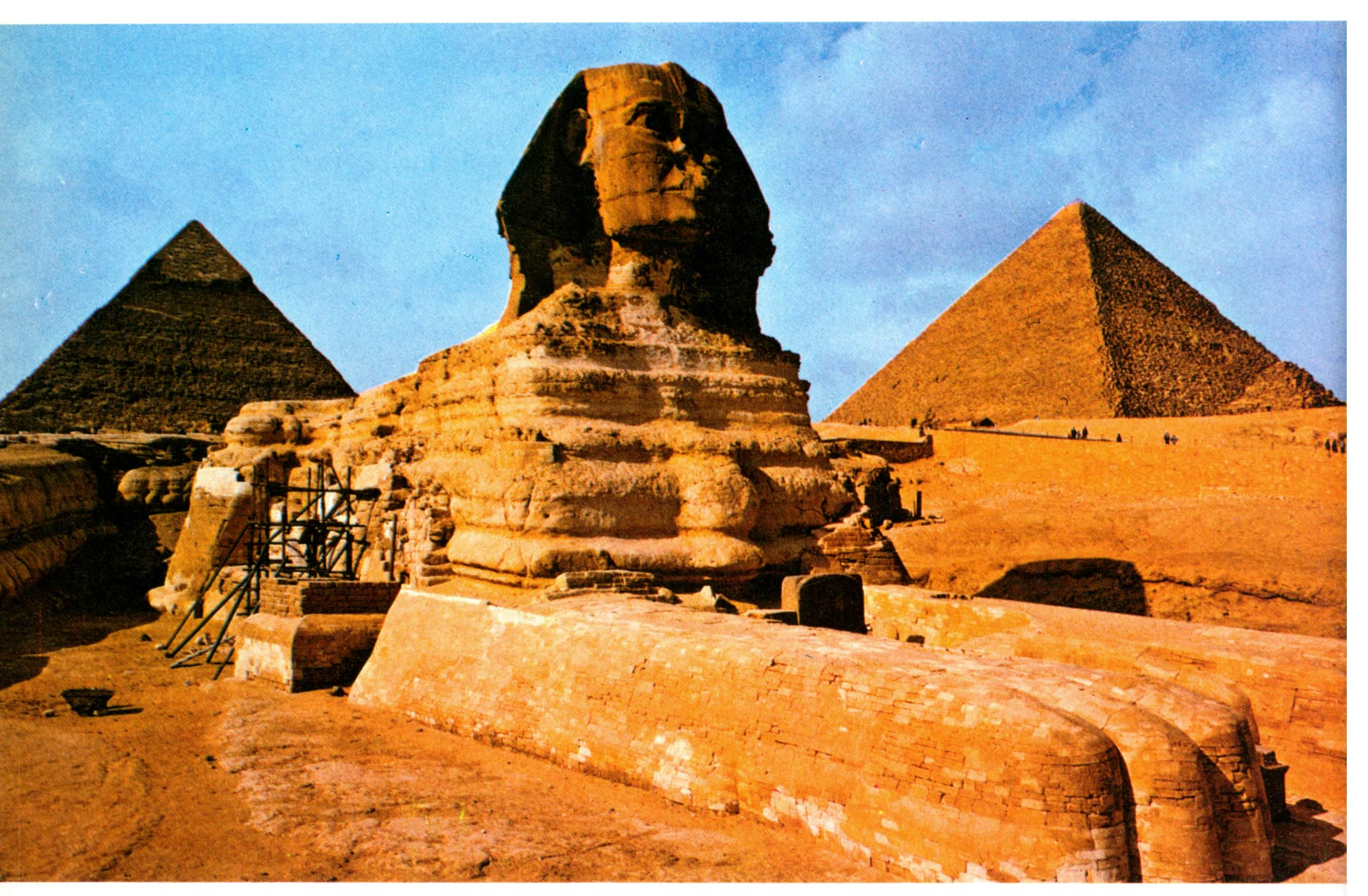
Más tarde, cuando el hombre aprendió a escribir, que es un buen sistema para no tener que acordarse de todas las cosas, ideó también un sistema para anotar los números. Una cabra, un palote; dos cabras, dos palotes; cinco cabras, una V; diez cabras, una X; cincuenta cabras, una L; cien, una C; quinientas, una D; mil, una M. Son los números romanos que te han enseñado. Los romanos se los copiaron a los egipcios, cambiándoles los signos y añadiéndoles los signos de 5, 50 y 500 que los egipcios no tenían. La ilustración muestra cómo era el sistema de numeración jeroglífica de los egipcios.

## OPERACIONES

Sigamos con el ejemplo del pastor de cabras. Cada vez que le nacía una o se la comía, la perdía o se la robaban, es decir,



*El papiro de Rhind, documento egipcio de gran valor, que muestra el sistema de numeración de los egipcios y constituye una prueba clara de la gran antigüedad de las matemáticas.*



*Gracias a sus grandes conocimientos matemáticos, los egipcios pudieron construir la colosal* Esfinge de Gizeh *y las pirámides de los faraones Keops, Kefrén y Mikerinos.*

cada vez que cambiaba el número de cabras que tenía, debía volverlas a contar, porque todavía no sabía sumar ni restar. Así que aprendió.

Te parece muy fácil sumar. Pero imagínate hacerlo sin saber que cuatro y ocho son doce, escribo un dos y llevo una. Es decir, imagínate una suma con números romanos de verdad. Y además escribiendo con un martillo o un pincel, o contando con piedras. No parece tan fácil, ¿verdad? Pero los antiguos sumaban y restaban y también multiplicaban y dividían. Por ejemplo, observa cómo multiplicaban los egipcios en el cuadro de la derecha.

## Las reglas de multiplicar

Para multiplicar nosotros utilizamos las tablas que nos han reseñado en la escuela. Compara la dificultad de hacerlo sin saberlas, tal como hacían los antiguos egipcios (cuadro adjunto).

| Nosotros | Los egipcios | |
|---|---|---|
| 13 | 1 | 7 |
| ×7 | 2 | 14 |
| | 4 | 28 |
| 91 | 8 | 56 |

el del 1 . . . . . . . . . . . . . . 7
el del 4 . . . . . . . . . . . . . . 28
el del 8 . . . . . . . . . . . . . . 56
13 . . . . . . . . . . . . . . 91

Los egipcios escribían los dobles del más pequeño en una columna, y apuntaban en la izquierda qué doble era.

De esta columna buscaban la combinación que sumaba el más grande: 1 + 4 + 8 = 13.

Por último sumaban los correspondientes números de la columna de la derecha.

*Cálculos primitivos*
Como puedes ver el asunto era complicado. Además, recuerda que sumaban de memoria, utilizando sus símbolos y esto los pueblos que los tenían, porque, por ejemplo, los griegos no tenían signos escritos para los números y utilizaban las letras del alfabeto. Como si dijéramos que la A es un uno, la B un dos y así sucesivamente. Para calcular utilizaban piedrecillas, que en griego se llaman cálculos; así que, cuando oigas que alguien está enfermo porque tiene cálculos en el riñón, es que tiene una piedra, no una división.
Como puedes suponer, todos los cálculos les resultaban muy difíciles y saber sumar y multiplicar representaba un mérito enorme. Pero, a pesar de esta dificultad, los antiguos egipcios supieron descubrir el número $\pi$ *(pi),* calcular la distancia de la Tierra al Sol y muchas otras cosas. Casi todas las escribieron en las dimensiones de la Gran Pirámide.

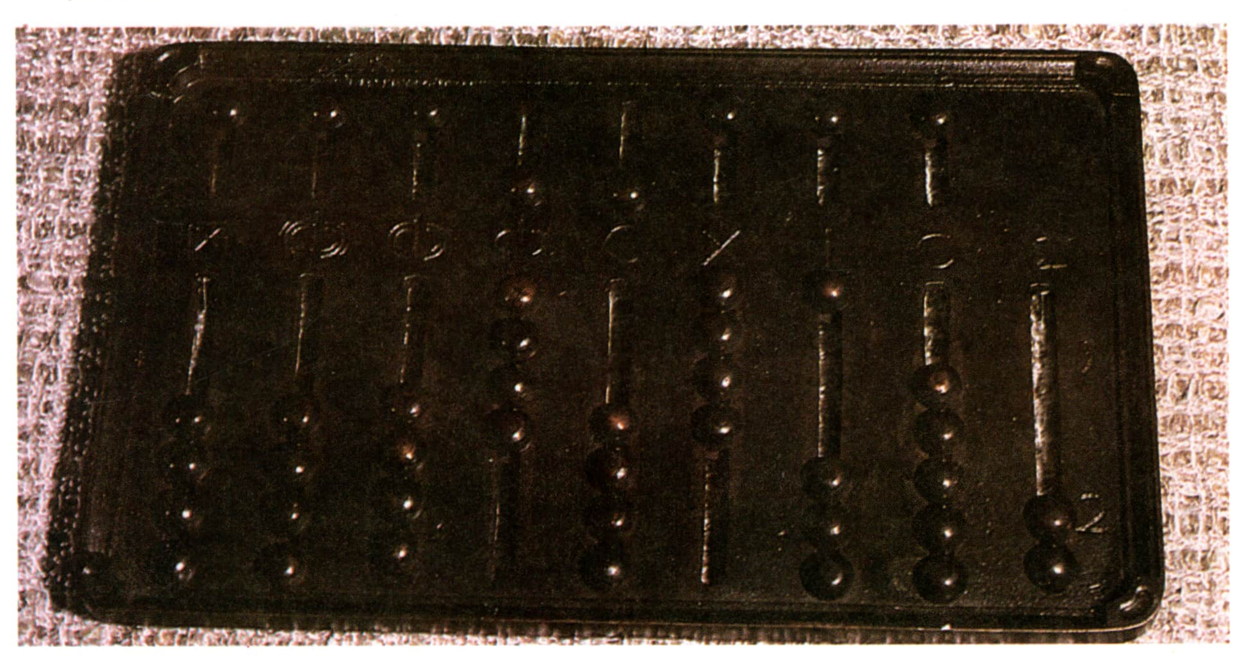
## El cero

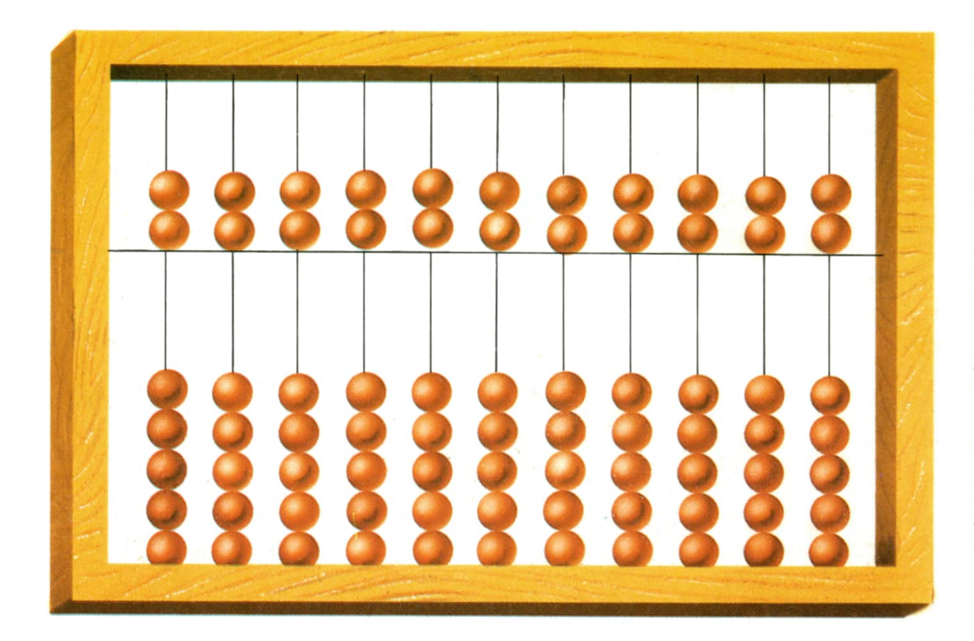
Si los métodos de cálculo eran tan complicados, se debía a que los signos que representan a los números eran todos diferentes. Para escribir mil escribían mil: M en números romanos. Nosotros escribimos 1.000, es decir: un millar, cero centenas, cero decenas, cero unidades. Parece algo tonto pero es el mejor sistema para calcular. Sólo hay diez cifras diferentes. Y la categoría que representan (unidades, decenas, centenas...) la toman por la posición que ocupan, por el orden que les corresponde. Fíjate la importancia que tiene el cero escrito. Lo inventaron los hindúes y los mayas. Pero sólo como signo escrito, porque como sistema de ordenar ya existía en los ábacos.

## Ábacos

Fueron, sin duda, la primera máquina de calcular. Su idea básica es ordenar los números por unidades, decenas, centenas. Mil se escribe moviendo la bolita de los miles. Es nuestro uno, cero, cero, cero, con la diferencia de que el cero no hay que escribirlo. Se indica como omisión.
Sumar es muy fácil: escribes un sumando y luego intentas escribir el siguiente. Si no te quedan bolitas le pones una a la columna de la izquierda y bajas todas las de la tuya. Es como si te diera el cambio. Hay unas reglas para agilizar la operación y otras para multiplicar y dividir.
Con práctica, el ábaco es tan rápido como una calculadora. Con la calculadora tienes que hacer exactamente lo mismo, sólo que

*Unos cinco mil años separan nuestras calculadoras de la primera máquina de calcular conocida: el ábaco. Abajo, ábaco romano.*





*El ábaco funciona ordenando los números por unidades, decenas y centenas. Mil se escribe moviendo la bolita de los miles. Es nuestro uno, cero, cero, cero, con la diferencia de que el cero no hay que escribirlo, pues se indica por omisión. Sumar es fácil: escribes un sumando y luego el siguiente. Si no te quedan bolitas, le pones una a la columna de la izquierda y bajas todas las de la tuya. Como si te diera cambio. Con práctica, el ábaco es tan rápido como la calculadora. Los chinos lo siguen utilizando aún hoy.*

has de escribir todos los ceros y además apretar la tecla -enter- después de cada sumando.
Los chinos los siguen utilizando todavía. Si vas a un restaurante chino, que sea chino de verdad, verás como la cuenta la suman con un ábaco. Y no creas que es para poner una nota exótica.

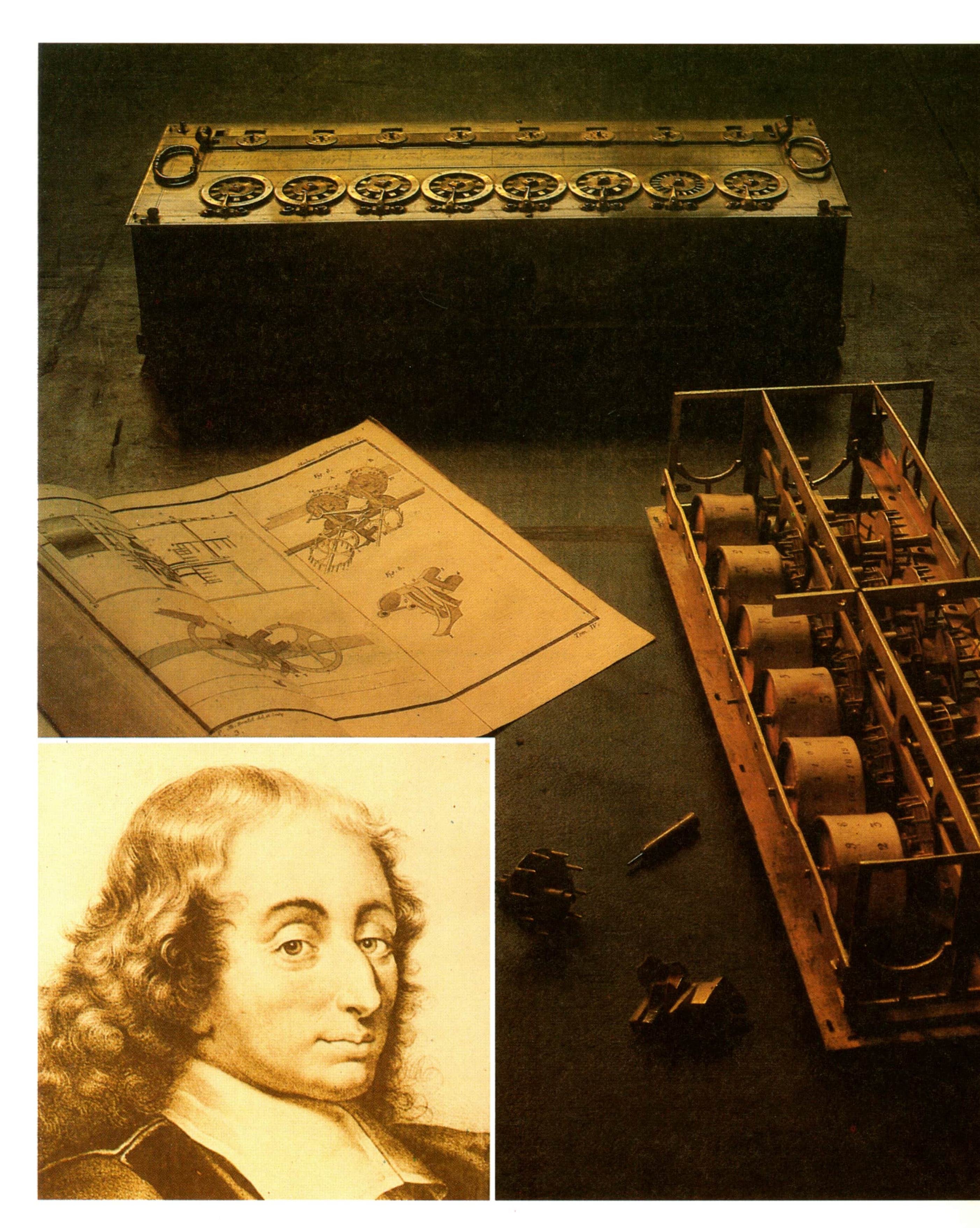
**Las calculadoras de Occidente**

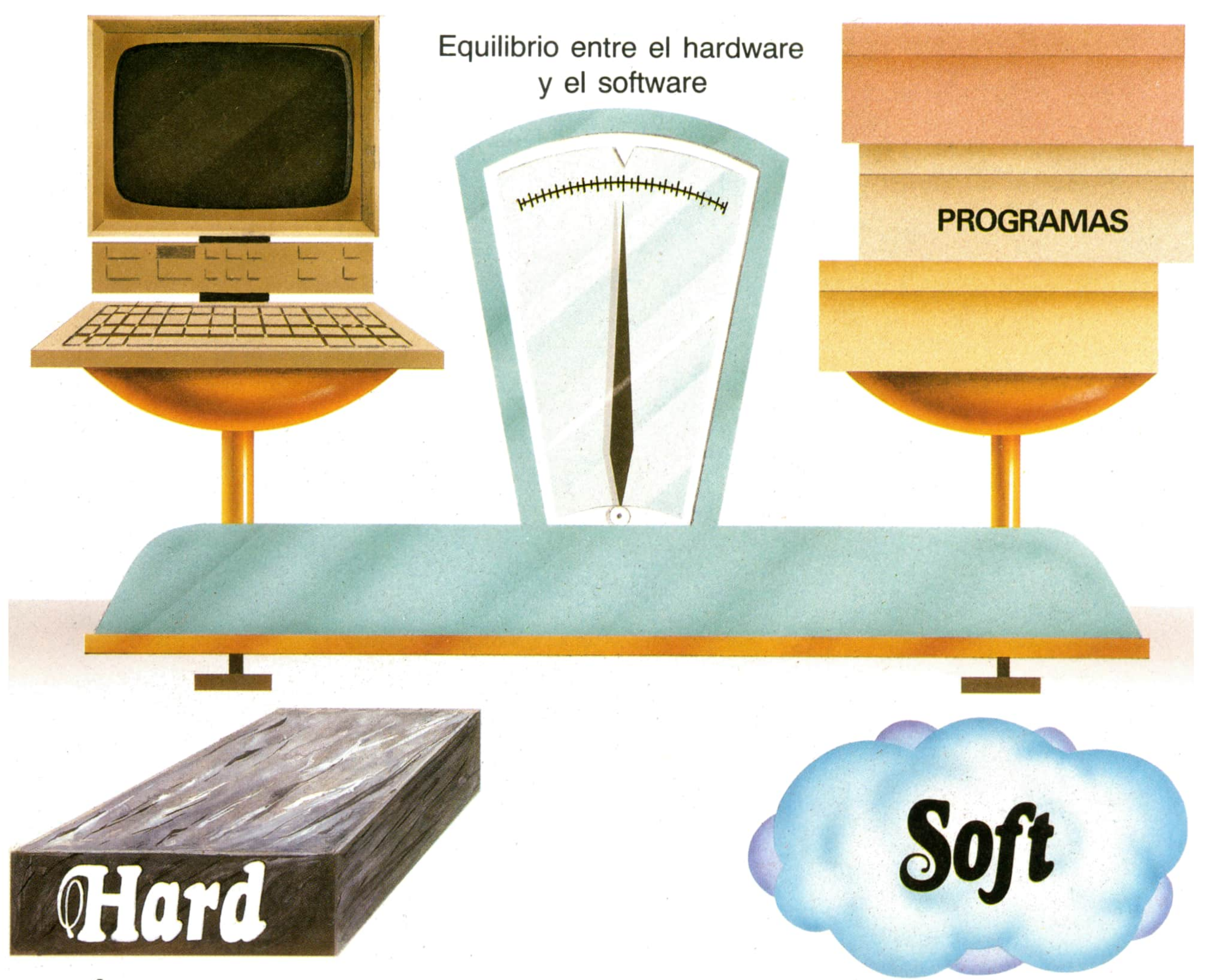
En nuestra sociedad occidental, los ábacos no tuvieron éxito. La gente sumaba con los dedos y lo escribía en un papel gracias a su sistema decimal. Pero como sabe todo el mundo, hacer una suma larga a mano es bastante cansado.
Claro que las sumas que hacían no acostumbraban a ser muy largas.
Los únicos que se veían obligados a hacerlas eran los recaudadores de impuestos. Y, como que la necesidad agudiza el ingenio, por aquí empezó la historia de las calculadoras.

## LA SUMADORA DE PASCAL

Una vez había un recaudador de impuestos que tenía un hijo que se llamaba Blaise Pascal, al que obligaba a ayudarle a sumar los impuestos que había cobrado. El chico, harto de sumar de memoria, decidió buscar una solución. Inventó una máquina para sumar. Podía haberse acordado del ábaco, pero prefirió servirse de unas ruedas y dibujarles por fuera los diez números. Les puso diez dientes, de modo que, cada diez movimientos, la rueda diera una vuelta completa y las unió de tal manera que, cuando una de las ruedas pasaba del nueve al cero, la rueda de su izquierda avanzaba una posición. Es lo mismo que lo del ábaco: si las bolas se acaban le pones una más a la columna de la izquierda y vuelves a situarlas todas en la columna que corresponde, pero la máquina de Pascal lo hacía de manera automática. Actualmente, el invento de Pascal se sigue usando en los cuentakilómetros de los automóviles.
La «Pascalina», como se llamó a esta primera máquina de sumar, fue el primer paso hasta las actuales computadoras. Por esta razón, Pascal es el nombre de uno de los lenguajes de programación.

*En la página contigua, la* pascalina *o sumadora de Pascal, con su complejo sistema de ruedas. En la parte superior, se observa la placa de la cubierta de la sumadora, en la que aparecen las posiciones numéricas elegidas, mientras que a la derecha se muestra la máquina en sección con el juego interior de ruedas y engranajes. Las ruedas de la* pascalina *llevan diez dientes, de modo que, cada diez movimientos, la rueda da una vuelta completa. Mediante el sistema de engranajes, cuando una rueda pasa del nueve al cero, la rueda de la izquierda avanza una posición. El invento de Pascal se usa actualmente en el cuentakilómetros de los automóviles. Abajo, la fotografía de Blaise Pascal.*






*El dibujo superior muestra el equilibrio que ha de existir entre las dos partes fundamentales del ordenador: el* hardware *y el* software. Hardware *es la parte material del ordenador, el conjunto de piezas físicas de que consta.* Software *es el conjunto de programas, la parte lógica de la máquina que nos permite realizar los trabajos de clase, por ejemplo.*

## SÍNTESIS

Repasemos lo dicho hasta ahora. Hemos hablado de una bombilla. Podemos decir que tiene una parte material: el cristal, el interruptor, los cables, los clavos que sujetan la línea a la pared, el portalámparas. Tiene también una parte que no es material, que no se puede tocar, pero que es tan necesaria como las piezas: el saber cómo se combinan. A esta parte le podemos llamar el «ingenio humano». Pero para que una bombilla se encienda y se apague hace falta otra cosa: saber cómo se utiliza.

Analicemos el ascensor. Analizar significa dividir algo complejo en sus partes más sencillas. Miremos si las tres partes en las que hemos dividido la bombilla nos sirven para estudiar el ascensor. En efecto, tiene una parte material: todos los aparatos, y una parte de ingenio: todas las conexiones. Estas dos partes son mucho más complicadas que en el caso de la bombilla. Y tiene una tercera parte: cómo se utiliza. Hay que saber, además, que para que el ascensor funcione se debe, antes que nada, cerrar las puertas y luego apretar el botón correspondiente. Y que no se debe salir en marcha, y que los menores de catorce años no lo pueden utilizar si no van acompañados. Lo advierte un letrero en todos los ascensores.

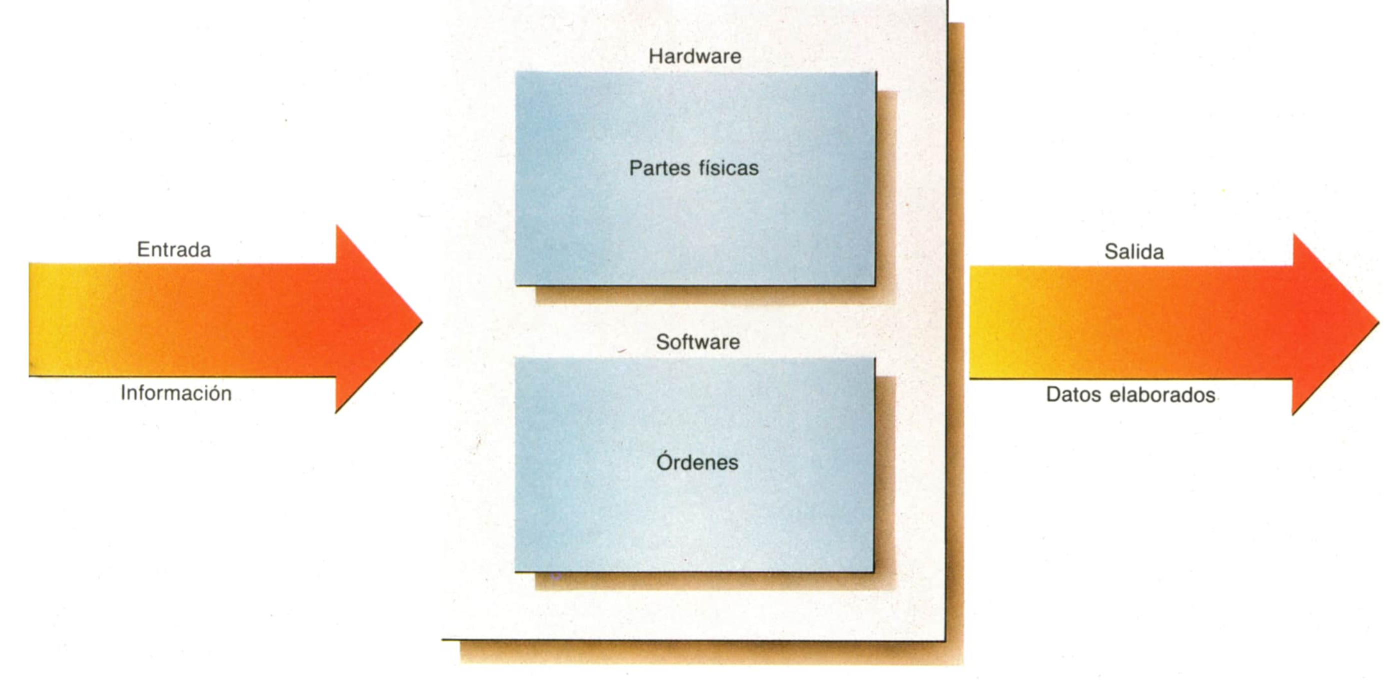
## HARDWARE Y SOFTWARE

Hemos explicado que, para que una máquina funcione, hacen falta tres cosas: las piezas, cómo están conectadas y cómo hay que utilizarlas. Podemos decir que la primera parte se encarga de todo lo que puede hacer la máquina. Encenderse y apagarse, como la bombilla. Subir, bajar y pararse como el ascensor. La segunda es cómo se le ha dicho que lo haga. Y la tercera cómo la hacemos funcionar.
Antes hemos dicho que el libro de física hablaba en «físico». Los lenguajes, no son más que convenciones de signos y palabras que se utilizan para entenderse. En la computación existe también un lenguaje: los lenguajes de programación, que nos permiten operar con las computadoras y entender sus mensajes. Por eso, para referirnos a conceptos y a cosas que en castellano normal no tienen definición nos las inventamos, aunque un tanto traducidas del inglés, todo hay que decirlo. Así a la primera parte en que hemos dividido las máquinas le llamamos HARDWARE, o sea al conjunto de piezas que la componen. A la segunda, o sea a cómo se consigue que la máquina cumpla su cometido le llamaremos SOFTWARE y a la tercera, es decir, al conjunto de instrucciones que hay que seguir para que la máquina obedezca a todo el mundo, le llamaremos MANUAL.

### Definición de «Software» y «Hardware»

Si buscas en un diccionario el significado de las palabras «hardware» verás que pone «quincallería», «chatarra». O sea lo que se puede tocar. «Software» es más difícil de definir. Es la parte lógica del asunto. SOFT significa blándo en inglés, y HARD, duro. El «manual» representa el conjunto de instrucciones que hay que seguir.
Mira la lavadora que tienes en tu casa. Tiene un «hardware»: un conjunto de hierro, con un motor, unos cables, una entrada de agua, y unos botones. Tiene su «software». Los programas. Lavado en frío, en caliente, con prelavado, centrifugado, toma de suavizante, dos aclarados, tres aclarados, etc. Y tiene un libro de instrucciones, el «manual», que permite al propietario de la lavadora elegir qué quiere que le haga su máquina.
Un ordenador es como una lavadora, sólo que los programas, el «software», se pueden ampliar a partir del que te dan cuando compras la máquina, siempre que sigas las normas que están en el libro de instrucciones, que se llama manual. Por eso al comprar un ordenador te dan tres cosas: una máquina, el «hardware». Un «diskette», el «software», y un libro de instrucciones o «manual».

*Esquema de funcionamiento del ordenador. La entrada de información en el* hardware *de la máquina se traduce, gracias al* software, *en datos elaborados.*



# EL ORDENADOR COMO UNA CALCULADORA

## LA MEMORIA

Hemos empezado definiendo el ordenador o computadora como una máquina que se acuerda de todo. Esto quiere decir que se trata de una máquina que tiene *memoria*. Esta propiedad es lo que distingue en principio a un ordenador de las otras máquinas. Pero esta no es la única diferencia.

Parece difícil imaginar que las máquinas se puedan acordar de las cosas. Pero es tan sencillo como decirle a una lavadora que se llene de agua, recoja el jabón, lave durante 10 minutos, se vacíe de agua, se vuelva a llenar, vuelva a lavar, y lo haga a la temperatura que le ordenamos. Y que después aclare tres veces; que se vacíe de nuevo y empiece a dar vueltas como una loca hasta que la ropa esté seca.

Otro ejemplo de memoria es el vídeo, que se pone a grabar a las ocho y media. O, cuando aprietas el botón del ascensor y se para en tu piso.

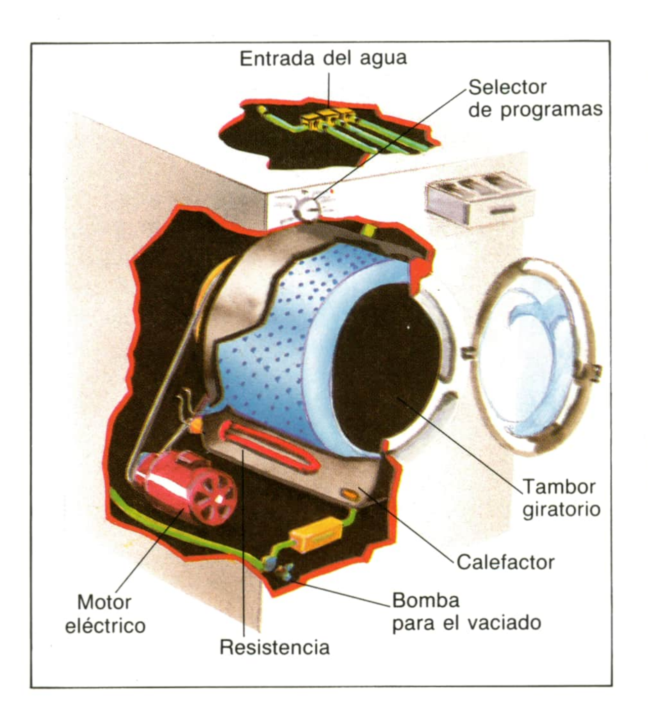
Pero ni a la lavadora ni al vídeo ni a los ascensores se les llama ordenadores, pero no porque tengan poca memoria, sino porque sólo sirven para una cosa: lavar, grabar, subir o bajar. En cambio, un ordenador puede servir para muchas cosas al mismo tiempo, aunque sin la memoria esto no hubiera sido posible.

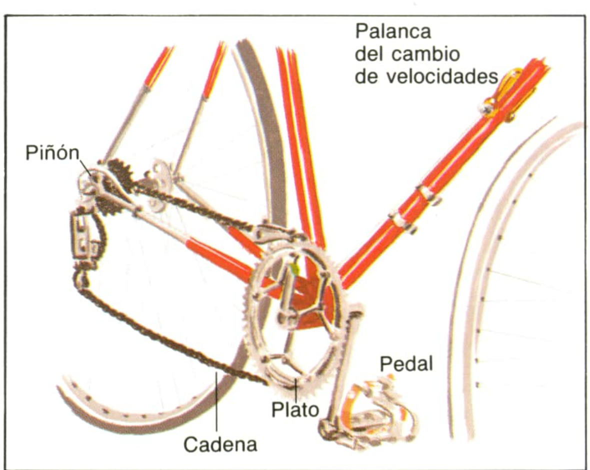
Y cuando tuvo la suficiente pudo llamarse ordenador.

### ¿Cómo se consigue que una máquina tenga memoria?

¿Te has fijado alguna vez en el cambio de marchas de un coche? Según donde está la palanca, el coche va más deprisa y hace menos ruido si el acelerador está siempre apretado con la misma intensidad. Éste es el ejemplo más sencillo de memoria. El coche se acuerda de a qué velocidad tiene que ir. Y se lo dices moviendo una palanca.

Para que entiendas el proceso, observa la figura inferior. Se trata del cuadro de una bicicleta. En primer lugar aparece el plato, que es una rueda dentada que transmite, gracias a la cadena, el movimiento imprimido por el ciclista en los pedales al piñón. El piñón es en las bicis corrientes una pequeña rueda dentada, aunque en la que aparece en la figura lleva incorporado



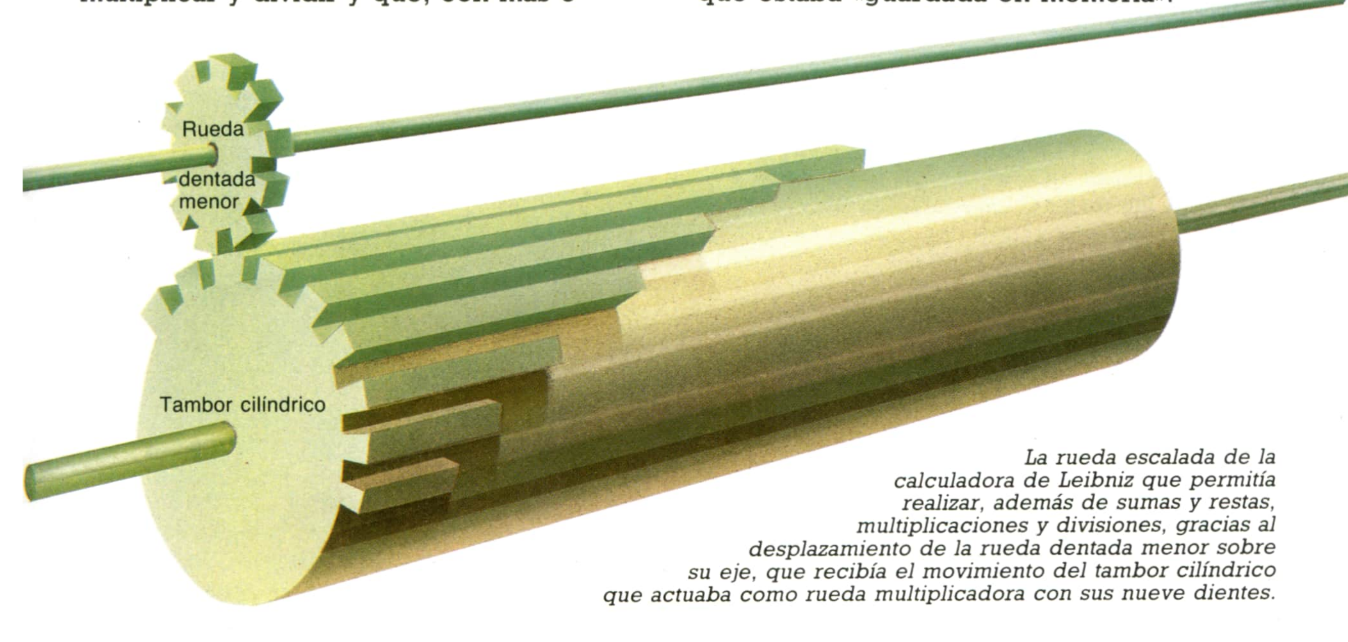

*Al igual que un ordenador, la lavadora (izquierda) posee una memoria. Pero, a diferencia de aquél, su memoria sólo sirve para una cosa: lavar; mientras que el ordenador puede realizar muchas cosas: jugar, hacerte los deberes, etc. La figura superior muestra el cuadro de una bicicleta, cuyo principio es análogo a la memoria del ordenador.*

el cambio de marchas que hace más complejo el sistema, pero que permite correr mucho más y hacer escalada, como en el *Tour*. Una auténtica bici de corredor. En principio, el funcionamiento de una bici corriente sería el siguiente. Cuando la rueda grande del plato da una vuelta, han pasado 45 dientes. Como la pequeña sólo tiene 15, para pasar 45 tiene que haber dado 3=45/15 vueltas. Es lógico, pensar que cuanto mayor sea la diferencia entre los dientes de las ruedas más vueltas dará la rueda de la bicicleta y, por lo tanto, más deprisa irá. Es también lógico, y el esfuerzo que tenemos que hacer lo demuestra, que, cuanto mayor sea la diferencia, más costará darle una vuelta a los pedales. Cuando se pensó que iría muy bien poder cambiar el número de dientes de la rueda pequeña, se inventó el cambio de marchas. Se unen tres ruedas y con una palanca se hace que la cadena de la bicicleta se mueva de una a otra. Podemos decir que hemos hecho una máquina que se acuerde de tres cosas. Poca memoria, pero algo es algo. Pero antes de los cambios de marcha se perfeccionaron las calculadoras.

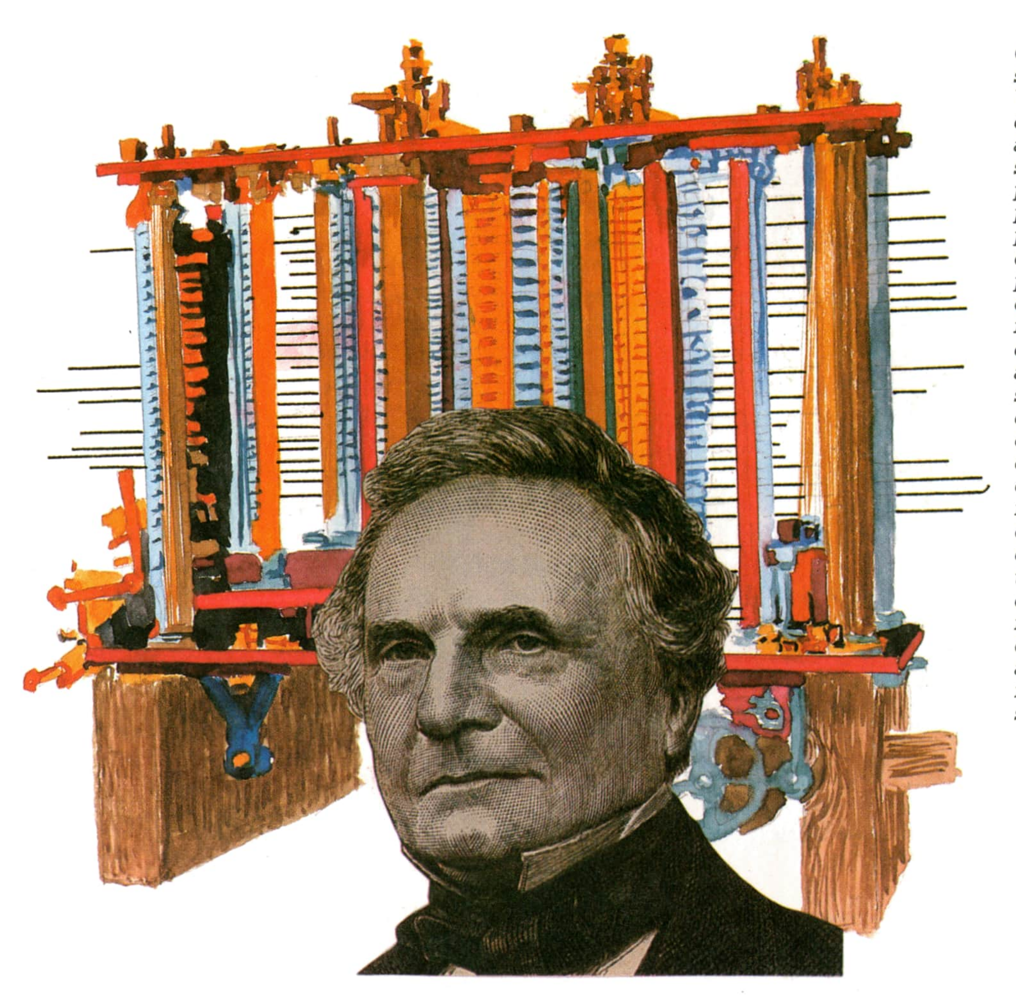
## Calculadoras

Despueś de la «Pascalina» se inventaron nuevas máquinas de calcular.
Leibniz, que era filósofo y matemático, construyó una máquina que además sabía multiplicar y dividir y que, con más o menos variantes, se siguió utilizando desde 1700 hasta hace unos veinte años. Se marcaba el multiplicando como en un ábaco, moviendo unas palancas hasta que se formaba el número. Se tomaba una manivela y se daban tantas vueltas como unidades tenía el multiplicador. Imagínate que queremos multiplicar por 153. Después de las tres primeras vueltas el número que habíamos marcado aparecía multiplicado por tres (en realidad se había sumado tres veces), se movía una palanca, que le decía a la máquina que ahora se iba a multiplicar por las decenas. Se le daban cinco vueltas a la manivela y el multiplicando se multiplicaba por 53. Por último, y otra vez con la palanca de selección, seguían las centenas. Una vuelta y ya teníamos el resultado.

*Calculadoras con memoria*
Cuando Pascal inventó su «Pascalina» no se contentó con su máquina de sumar, porque pensó que estaría muy bien que su aparato supiera hacer sumas parciales. De hecho, le bastaba con tener dos pascalinas y pasar manualmente los resultados de una a la otra y poner luego a cero la primera. Conseguir que las sumas parciales pasaran de una a otra automáticamente ya era más complicado, pero Pascal lo logró con un cambio de marchas o sea con una palanca que conectaba una serie de ruedas y desconectaba otra. De la serie de ruedas dentadas desconectada podemos decir que estaba «guardada en memoria».




*La rueda escalada de la calculadora de Leibniz que permitía realizar, además de sumas y restas, multiplicaciones y divisiones, gracias al desplazamiento de la rueda dentada menor sobre su eje, que recibía el movimiento del tambor cilíndrico que actuaba como rueda multiplicadora con sus nueve dientes.*



*Charles Babbage y su máquina analítica. Verdadero precursor del ordenador, se adelantó en casi un siglo a su época y no pudo poner en práctica el invento, porque no se conocían aún las propiedades de la electricidad. Lo realmente asombroso de la máquina analítica no radica en su gran capacidad de realizar operaciones diversas, sino que está organizada del mismo modo que un ordenador actual. En efecto, dispone de un mecanismo de entrada de datos, de memoria, unidad de control, unidad aritmético-lógica y mecanismos de salida.*

*El pionero de la conmutación: Charles Babbage*

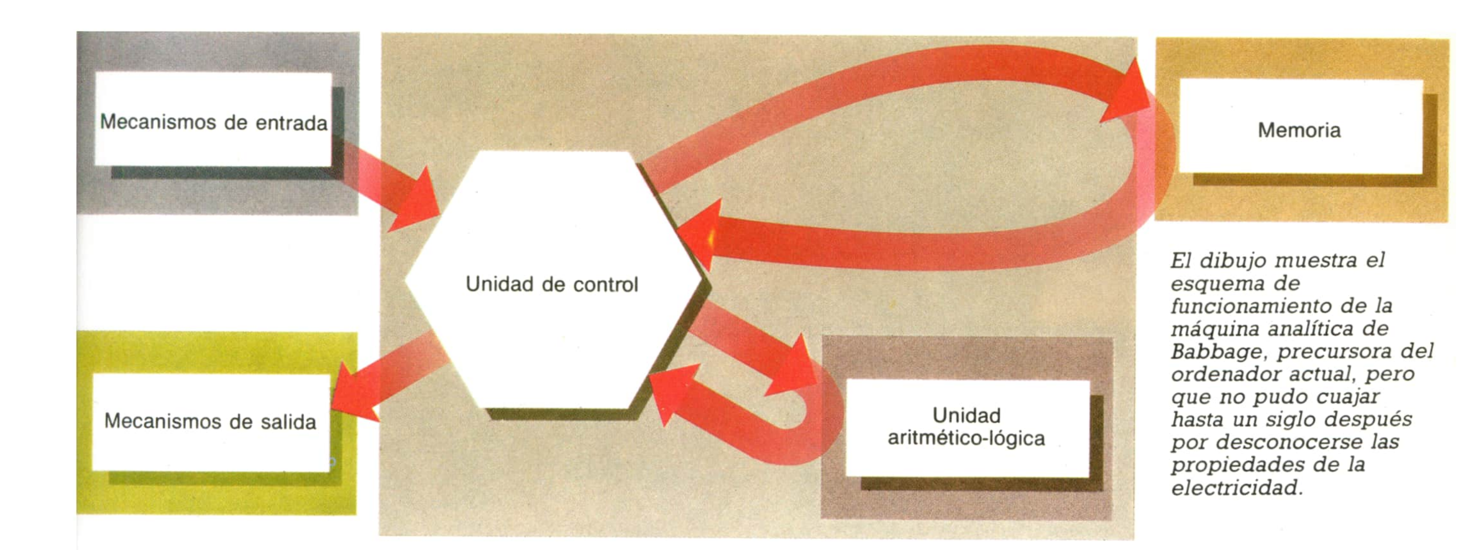
Desde el invento de Pascal hasta los ordenadores actuales hay un largo camino y muchos inventores, pero al primero que se le ocurrió pensar en una máquina que supiera calcular de verdad, es decir, que pudiera hacer multiplicaciones, divisiones, que acumulara sumas de los productos y suma de las sumas, que restara productos de las divisiones y todo lo que se le ocurriera al usuario, es decir, lo que ahora hace una calculadora de bolsillo fue Charles Babbage (1792-1871). Fue lo que se dice un invento «a lo grande», que no fue bien entendido en su época y su soñada «máquina analítica», como él la había bautizado, se quedó en proyecto. Sus ideas tuvieron que esperar a que el mundo cambiara lo suficiente. En realidad, sólo le faltaba utilizar la electricidad, pero Babbage no conocía sus propiedades. Ni nadie por entonces.

## Partes de una calculadora

A pesar de su fallo en los resultados conseguidos, Babbage fue el primero que sentó las bases de la fabricación de un ordenador. Se dio cuenta de que las calculadoras se deben dividir en tres partes muy diferenciadas. Primero se deben entrar los datos, después hay que hacer las operaciones y luego hay que enseñar el resultado. Tanto lo acertó que los ordenadores actuales siguen con su misma división. Vamos a ver con detalle cada una de estas partes.

*Salidas*

Es la parte más importante, la constituyen los resultados. Es el motivo por el cual hay calculadoras. Porque aunque calculen lo que sea y tan rápido como quieran, si no nos enseñan el resultado, no nos sirven para nada. Al principio fue lo más sencillo de diseñar: una rueda dentada con los números escritos en su parte exterior.




*El dibujo muestra el esquema de funcionamiento de la máquina analítica de Babbage, precursora del ordenador actual, pero que no pudo cuajar hasta un siglo después por desconocerse las propiedades de la electricidad.*

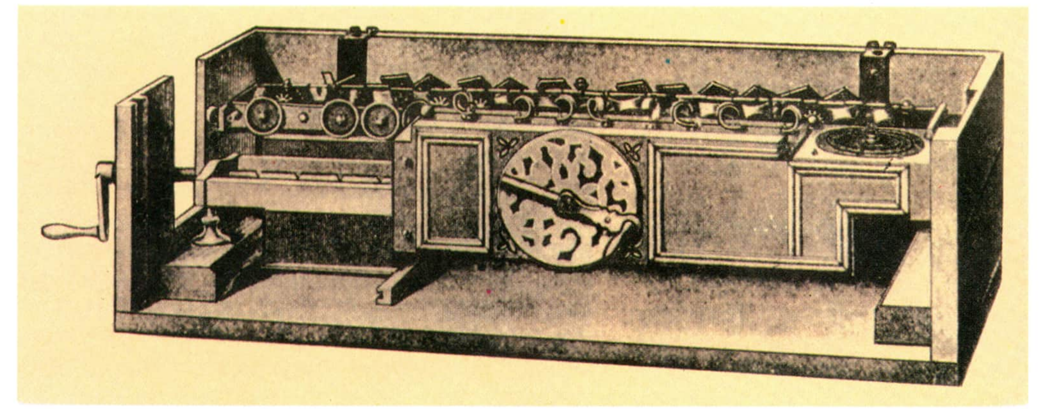
Podríamos decir que es el sistema que tienen las máquinas de comunicarse con nosotros. Actualmente, las salidas más frecuentes son las pantallas y las impresoras.

*Entradas*
Es una parte comparable a la gasolina de los coches: el combustible. Las máquinas funcionan porque se las alimenta con datos. En las primitivas calculadoras se hacía girando una manivela para mover las ruedas. En realidad, se trata del lenguaje que debemos emplear para que nos entiendan. Actualmente y en la mayoría de los casos, se trata de apretar botones o teclas.

## Cálculos

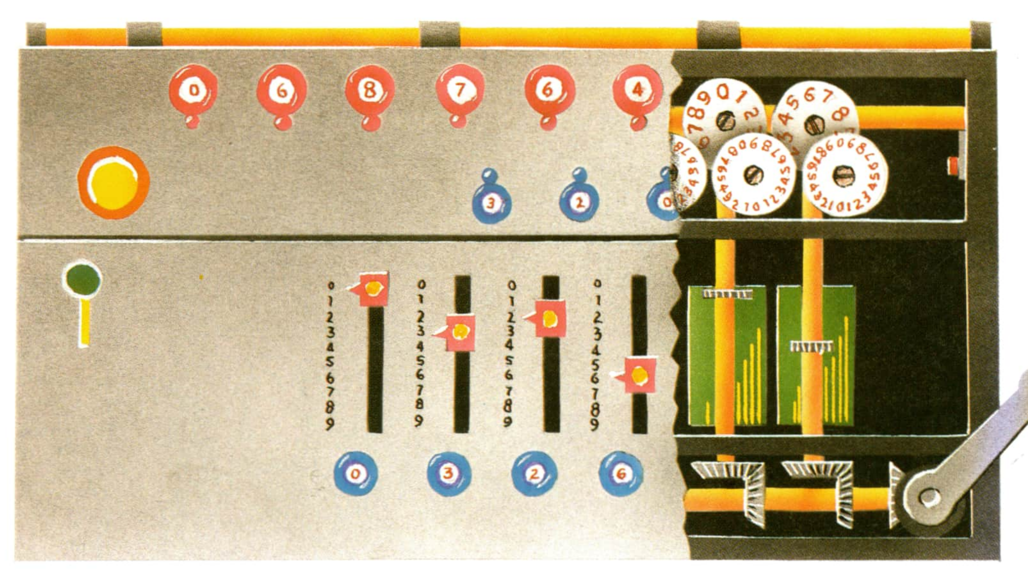
Es la parte más complicada, tanto que hay que dividirla a su vez para estudiarla mejor.





*Arriba, retrato de Leibniz, filósofo y matemático alemán, creador de la «calculadora universal» (abajo), como la bautizó el propio Leibniz, dotada de un ingenioso mecanismo, la rueda escalada, que permitía multiplicar y dividir.*

Esquema de la máquina de Odhner



*Esquema de la máquina de Odhner, calculadora que funcionaba a base de ruedas con perno. La rueda era un disco fijo a una manivela y una parte giratoria. La capacidad de cálculo de la máquina dependía del número de ruedas con pernos.*

En la pascalina los cálculos sólo eran las ruedas dentadas, sabiamente engranadas entre sí. Pero en la máquina de multiplicar hay además una palanca que cambia las unidades por las decenas o por las centenas. Tenemos ya dos subdivisiones: una parte que calcula, las ruedas dentadas, y otra parte que dice lo que hay que calcular y cómo hay que hacerlo: la palanca de cambio. Actualmente, a la primera parte se la llama *unidad aritmético-lógica* y a la segunda *procesador.*

## Memoria

Cuando a las calculadoras se les añadió la memoria, la parte correspondiente a los cálculos se tuvo que ampliar. Además de las ruedas dentadas que calculan, se le añadieron las que están en memoria. A éstas se les llama actualmente la *memoria*. El procesador pasó de ser la sencilla palanca de las multiplicadoras a un complicado sistema de embragues y palancas para poner a trabajar una u otra rueda. Tan complicado era que sólo se resolvió a nivel teórico, es decir, que no se consiguió que funcionara. Ya se ha dicho que Babbage no sabía para qué servía la electricidad. Pero la historia de los ordenadores continuó y sus distintas partes fueron evolucionando.

## La entrada de datos

De las tres partes de un ordenador fue la que primero se desarrolló. Sin duda, porque era lo único que podía hacerse antes del empleo de la electricidad. El primero que lo logró fue Joseph Marie Jacquard (1752-1834). Ideó cómo decirle a una máquina de una manera muy sencilla, lo que tenía que hacer. Supongo que se fijó en las cajas de música. ¿Has desmontado alguna? Es tan sencillo que da rabia no haber pensado antes en hacerlo. Se trata de un cilindro con unos pequeños pinchos dispuestos sin ningún orden aparente. Tiene también unas lengüetas afinadas que rozan el cilindro. Si se hace girar el rodillo (y basta con darle cuerda), un pincho hace subir la lengüeta y luego la suelta, ésta suena en su nota. Y luego otra, y otra. Si hay bastante espacio entre dos pinchos consecutivos, la música es lenta; si están más juntos, suena más rápida. Si tres suenan a la vez es un acorde. Y así se puede hacer música.

*Codificación*

El inventor de la caja de música merecía el título de socio fundador del club de los programadores. Porque se pasó mucho tiempo haciendo el programa, es decir, situando los pinchos hasta que la melodía sonara bien. Claro que su máquina sólo tenía entradas y salidas. El proceso era la simple aplicación de la ley de la elasticidad de las lengüetas. Pero esto no le quita ningún mérito. Fue el primer codificador de la historia de las computadoras.

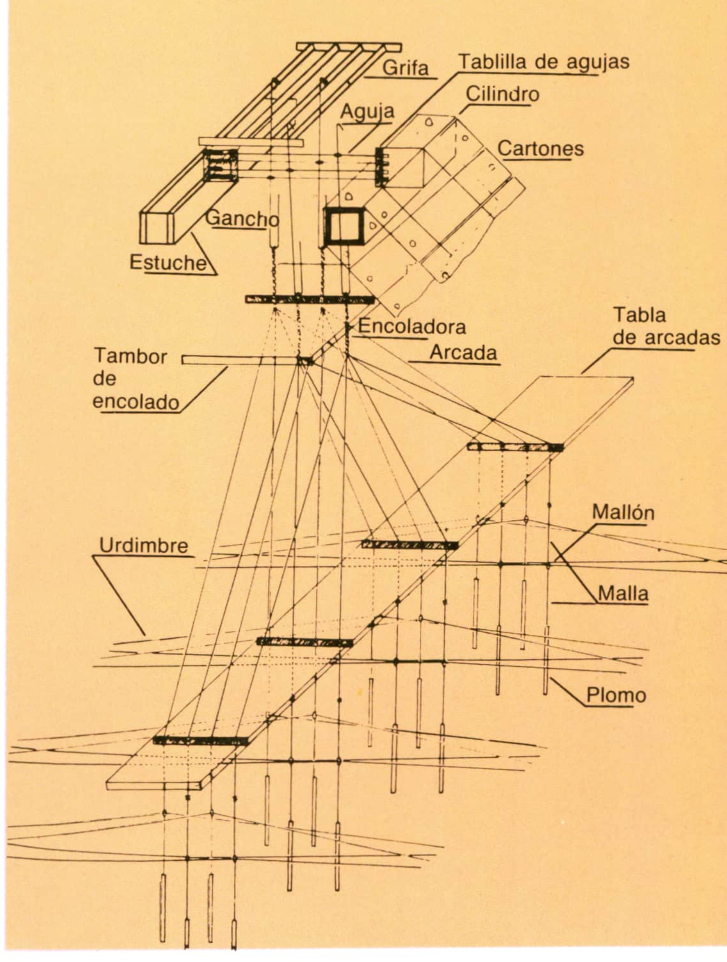
Codificar quiere decir traducir a un «idioma» raro. Cuando traduces del o al

francés no piensas que codificas, porque cambias «gracias» por *merci*. Pero es lo mismo. Lo que pasa es que lo de código suena más bien a secreto, a idioma extraño; al idioma de los espías.

*Tarjetas perforadas*
Jacquard le dio la vuelta a la caja de música. En vez de poner pinchos sobre un rodillo, le hizo agujeros a un cartón. Acababa de inventar la tarjeta perforada. Él no lo sabía, sólo quería que una máquina supiera hacer un tapiz, es decir, qué colores de hilos tenía que elegir en cada momento. Imagínate una serie de agujas enhebradas cada una con un hilo de color diferente. Si las envías todas hacia un cartón perforado, sólo pasará la que encuentre un agujero. Ya has seleccionado un color. Sólo haciendo un agujero al cartón le has dicho a la máquina que quieres que el hilo sea de color azul. Con la ventaja de que una vez tienes los cartones hechos, la selección del color será siempre la misma. Es tan sencillo como las plantillas para dibujar que, siguiendo el contorno, te queda el dibujo. Y siempre el mismo.
De la distribución de pinchos sobre un rodillo o de los agujeros en un cartón podemos decir que son una codificación, es decir, algo que sólo entiende aquél a quien va destinado. La caja de música y el telar eran ingenios mecánicos capaces de entender, mejor sería decir «leer», los mensajes transmitidos por los pinchos sobre el rodillo o los de las tarjetas perforadas. Por eso se habla hoy de la *lectura de los datos*.

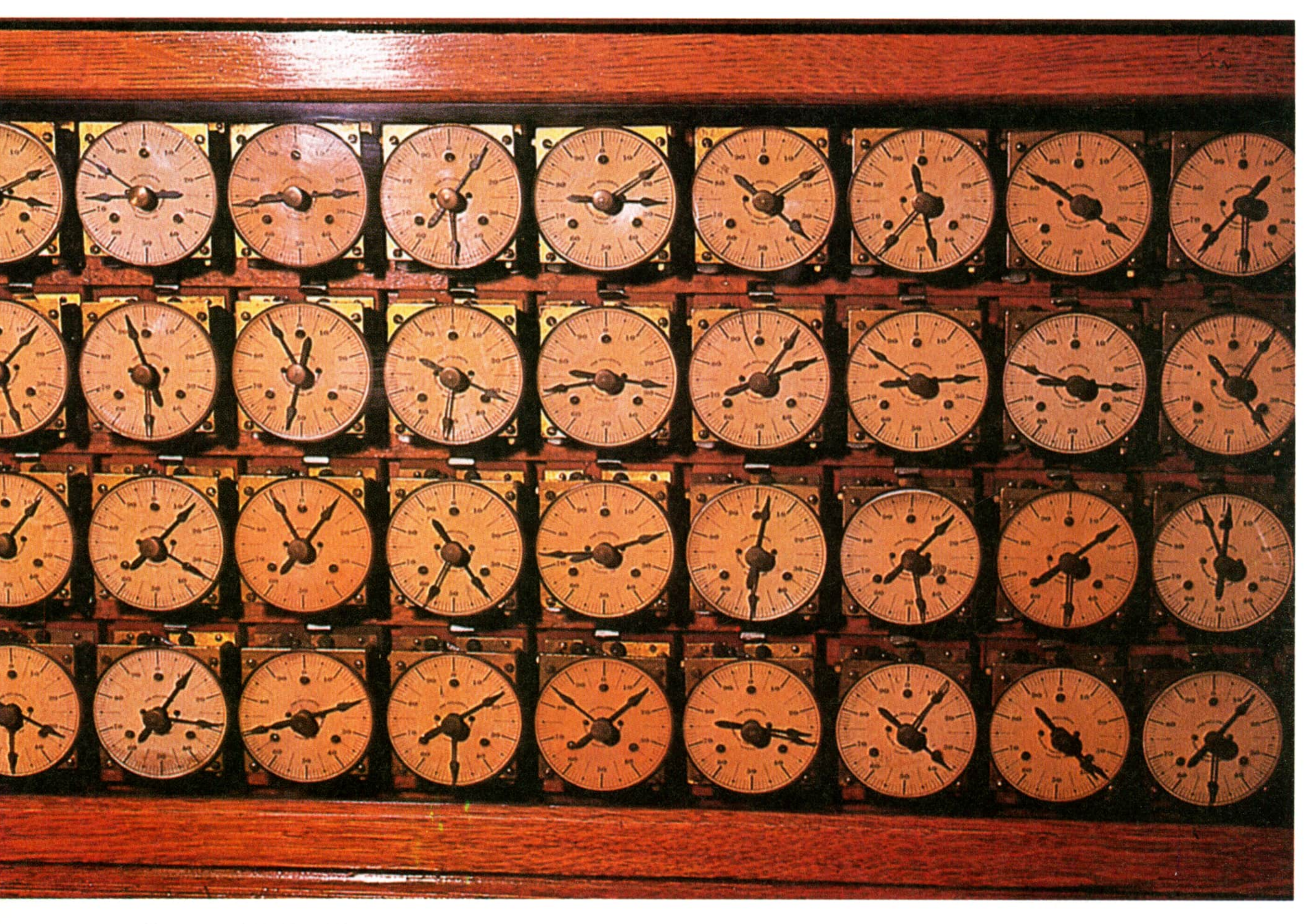
## La electricidad

A pesar de que Babbage incorporó a su máquina analítica una entrada de datos con una tarjeta perforada, la complicación que representa jugar con ruedas dentadas no le permitió desarrollar su máquina. Pero, cuando se descubrieron las propiedades de la electricidad, las cosas cambiaron. Ya hemos visto que con los relés se puede contar. Y sumar no es más que ir contando. Sólo había que tomar la máquina de Babbage y hacerla eléctrica. De eso se ocuparon dos personas; una se ocupó del problema de modo teórico, mientras que la otra buscó su lado práctico.




*Esquema del telar sistema Jacquard. De las tres partes de que se compone el ordenador, la que primero se desarrolló fue la entrada de datos, gracias al invento del telar mecánico del francés Joseph Marie Jacquard. La máquina tejía siguiendo un patrón o programa de trabajo escrito en unas tarjetas perforadas, que permitían reproducir sobre la tela dibujos formados por colores diferentes. Era el primer paso para codificar las instrucciones.*

*Ada Byron.*
Fue la teórica. Conoció a Babbage, le gustaron sus ideas y supo perdonar sus fallos, porque comprendió que la culpa era de la energía mecánica, la única que Babbage conocía. Al saber de lo que era capaz la energía eléctrica, no se cansó de decir que Babbage tenía razón. Finalmente consiguió que le hicieran caso. Por esta razón ADA es el nombre de un lenguaje de programación que utiliza mayormente la NASA.



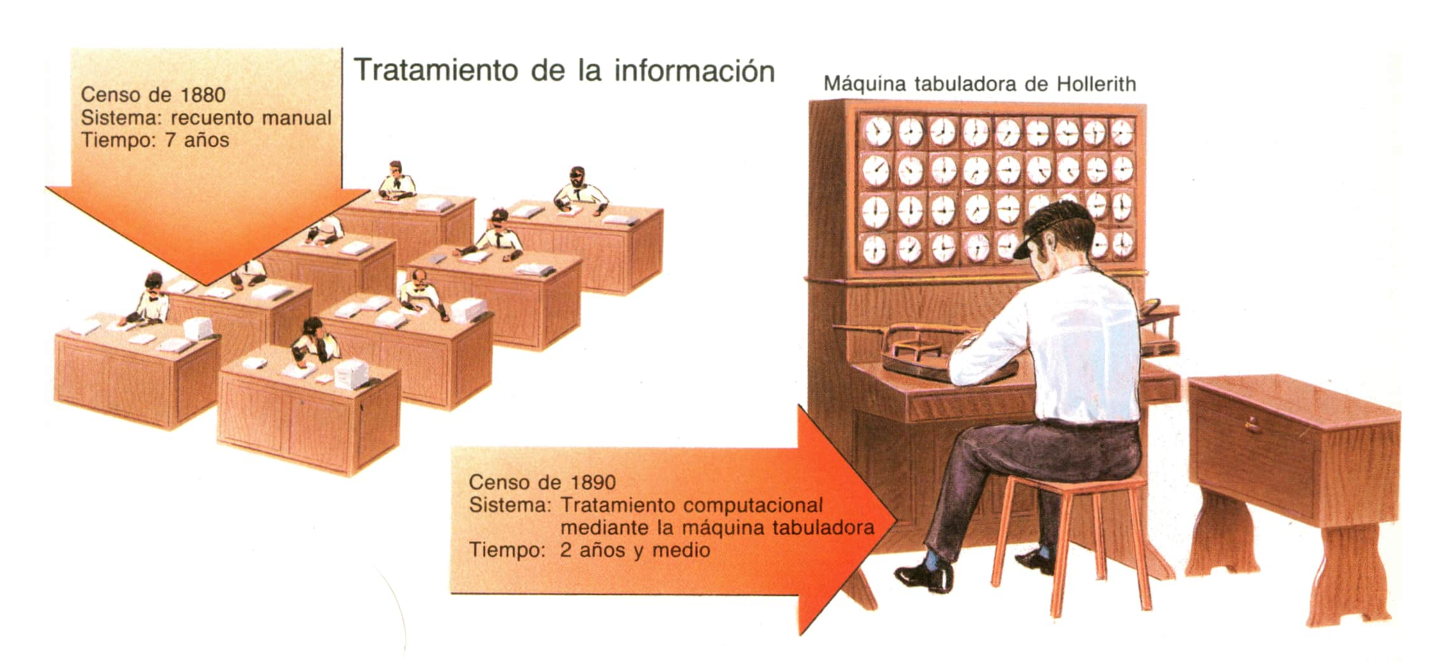
La máquina tabuladora de Hermann Hollerith, que realizó el censo de 1890 por encargo de la Oficina Federal del Censo de Estados Unidos en el tiempo récord para aquella época de dos años y medio.

*Herman Hollerith*
Fue el práctico. El que realizó la máquina. Hizo un monstruo que calculó el censo de los Estados Unidos (es decir sumó hasta 63 millones) en tan sólo dos años y medio. Parece de risa, pero la última vez que lo habían hecho (y entonces sólo habían contado hasta 55 millones) tardaron siete años.
A partir de aquí la historia de las máquinas se disparó. Hollerith fundó una empresa, la Tabulation Machine Company en 1896, que en 1924 se llamó International Bussines Machines (IBM).

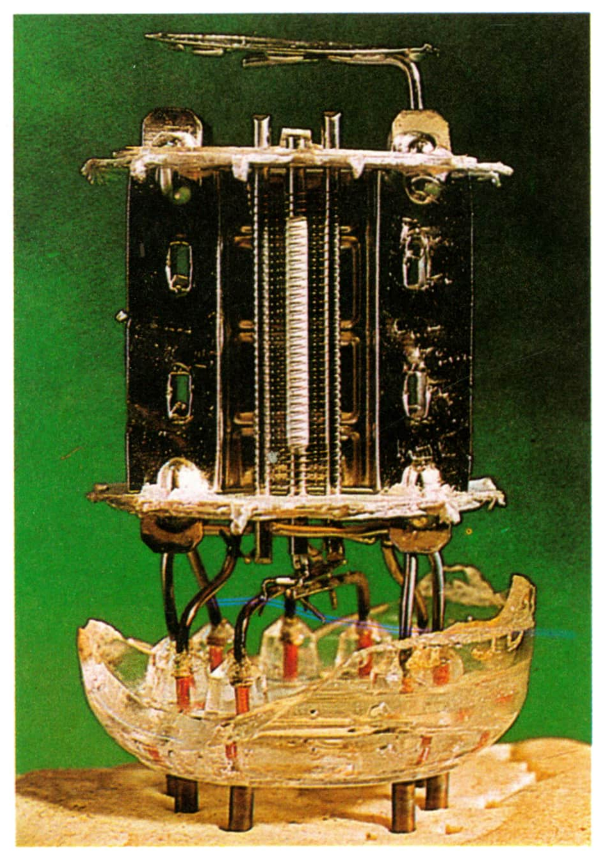
*El primer ordenador*
Así nació el MARK 1, un aparato de 5 toneladas, 5.000 relés, 700.000 mil elementos móviles y tres millones de conexiones eléctricas, que hacía una multiplicación de dos números de diez cifras en tres segundos y se podía acordar de 72 números de 23 cifras decimales. En realidad, tenía menos memoria de la que había pensado Babbage, pero era mucho más rápida.
Un relé tiene la pega de que es demasiado grande y aparatoso. Tiene partes móviles que, naturalmente, se estropean. Y su única utilidad para las computadoras consiste en conectar y desconectar circuitos. Y para la electricidad que debe circular son demasiado potentes. Por eso las calculadoras eran tan gigantescas y tan lentas. Cuando se inventaron las válvulas de vacío, los relés desaparecieron de las computadoras.

## Válvulas de vacío

Tal vez hayas visto una radio antigua. Está llena de una especie de bombillas extrañas que dan muy poca luz y que deben calentarse antes de que la radio empiece a




*El dibujo superior muestra el progreso conseguido entre el sistema de recuento manual del censo de población, llevado a cabo en 1880 en Estados Unidos (que duró 7 años) y el realizado en 1890 por Hollerith con su tabuladora en tan sólo dos años y medio. La tabuladora de Hollerith estaba compuesta de un lector de tarjetas, un contador, un clasificador y un aparato de tabular.*



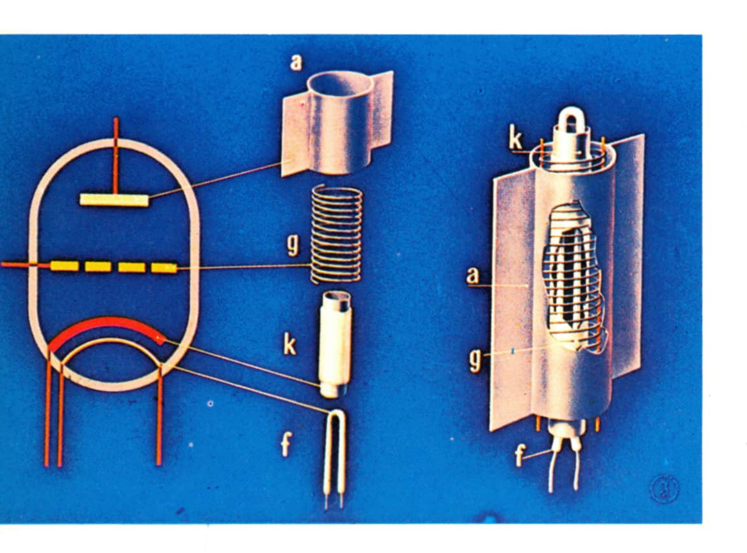
*Interior de un diodo o tubo de vacío. Las válvulas de vacío se basan en la propiedad de los electrones de pasar del polo negativo al positivo. Las válvulas de vacío fueron utilizadas en la primera generación de ordenadores, porque aventajaban a los relés en que ocupaban mucho menos espacio y se estropeaban menos.*

sonar. Estas válvulas son muy simples. Se basan en la propiedad de los electrones de pasar del polo negativo al positivo. Ya sabes que los electrones son la parte de los átomos que al moverse producen la corriente eléctrica.

Coloquemos ahora entre los dos polos una especie de reja y conectémosla a la red. Si la reja no tiene corriente los electrones pasarán y por el polo positivo saldrá corriente. Si, por el contrario, damos corriente a la reja negativa, por ejemplo, los electrones no podrán pasar. Porque debes saber que la corriente va de negativo a positivo. Como puedes ver, esta válvula hace la misma función que los biestables, se enciende y se apaga, pero con dos grandes ventajas: necesita mucha menos corriente y además no tiene partes móviles, o sea que se estropea menos.

## Calculadoras electrónicas

Cambiando las ruedas dentadas de Babbage por relés se construyó el MARK 1, cambiando los relés por válvulas se fabricó el ENIAC. El ENIAC tenía 17.000




*Arriba, esquema y diversas partes del triodo, válvula de vacío —también conocida en el argot electrónico como tubo termiónico—. Cuatro son los elementos de la válvula de vacío: el filamento **(f)**; el cátodo **(k)** —que es el electrodo negativo—; la rejilla **(g)** —cuya misión es la de variar el campo magnético en el interior del tubo, de modo que los electrones circulen entre los polos negativo y positivo—; y el ánodo **(a)**, que es el electrodo positivo. Los triodos fueron sustituidos por los transistores, provistos de diminutos chips de silicio (derecha).*




válvulas, pesaba 30 toneladas, gastaba como 1.000 lavadoras, pero hacía la misma multiplicación que el MARK 1 en 0,003 segundos, es decir era mil veces más rápida. También sabía hacer raíces cuadradas y decidir entre dos números cúal era el más grande. Que, como veremos más tarde, es la mayor muestra de inteligencia que nos ofrecen los ordenadores.

Las válvulas eran todavía demasiado grandes, gastaban demasiada electricidad y se fundían frecuentemente. Tanto que había un equipo que sólo se encargaba de cambiar las estropeadas. En suma, la informática todavía estaba en pañales. Era preciso reducirlo todo: el tamaño, el gasto y el precio del diseño y montaje de todo. Esto lo consiguió el transistor que viene a ser lo mismo que una válvula pero en microscópico.

### Ordenadores transistorizados

La aparición del transistor en 1956 significó el despegue definitivo en la fabricación de ordenadores o computadoras. El transistor es mucho más pequeño (lo que abarata su coste), consume mucha menos energía eléctrica (lo que reduce los costes de funcionamiento), se estropea raramente (lo que disminuye los gastos de reparación) y, además, permite fabricarlos ya montados. Es lo que se llama circuito integrado o en el lenguaje informático *chip,* esa especie de arañita que muestran la ilustración, superior y el dibujo de la derecha. Piensa que, si tiene este tamaño, es porque hace falta espacio para colocarle las patas. Porque el espacio que ocupan los circuitos integrados es mucho más pequeño. Al transistor se debe la presencia habitual de computadoras por todas partes.

## SISTEMA BINARIO

### Cómo cuentan las calculadoras

La principal dificultad que tenía la máquina de Babbage consistía en que, para guardarse un número en memoria, había que disponer de una serie de ruedas dentadas en una determinada posición. Esto ocupaba evidentemente demasiado espacio. Además, operar con ellas

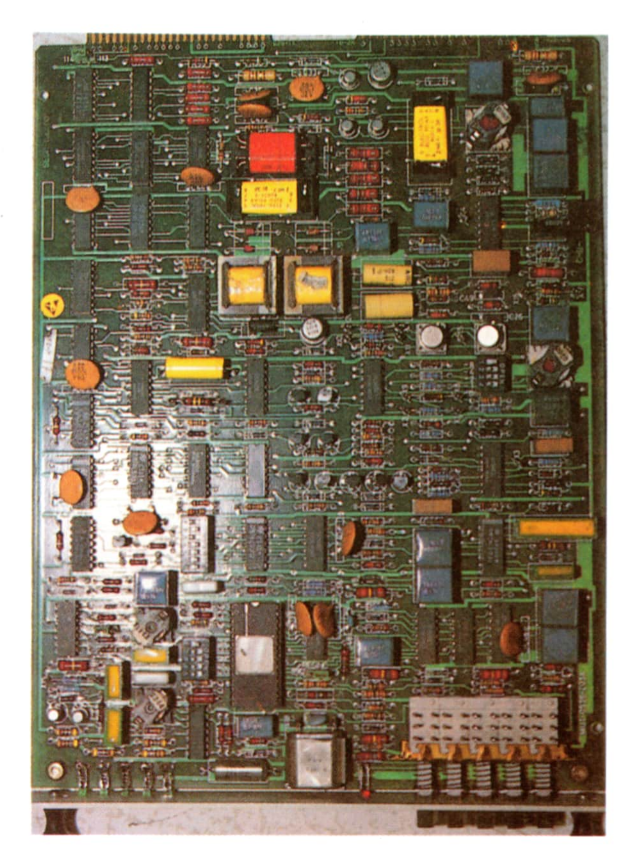
## Circuito integrado

Referencia de posición

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

*Los circuitos integrados son las unidades elementales de los ordenadores. Se conectan entre sí para formar el circuito impreso de la máquina a través de patillas, numeradas en el orden que se indica en el dibujo. Poseen unas muescas en su parte superior que permiten conocer la posición que deben ocupar en el montaje.*

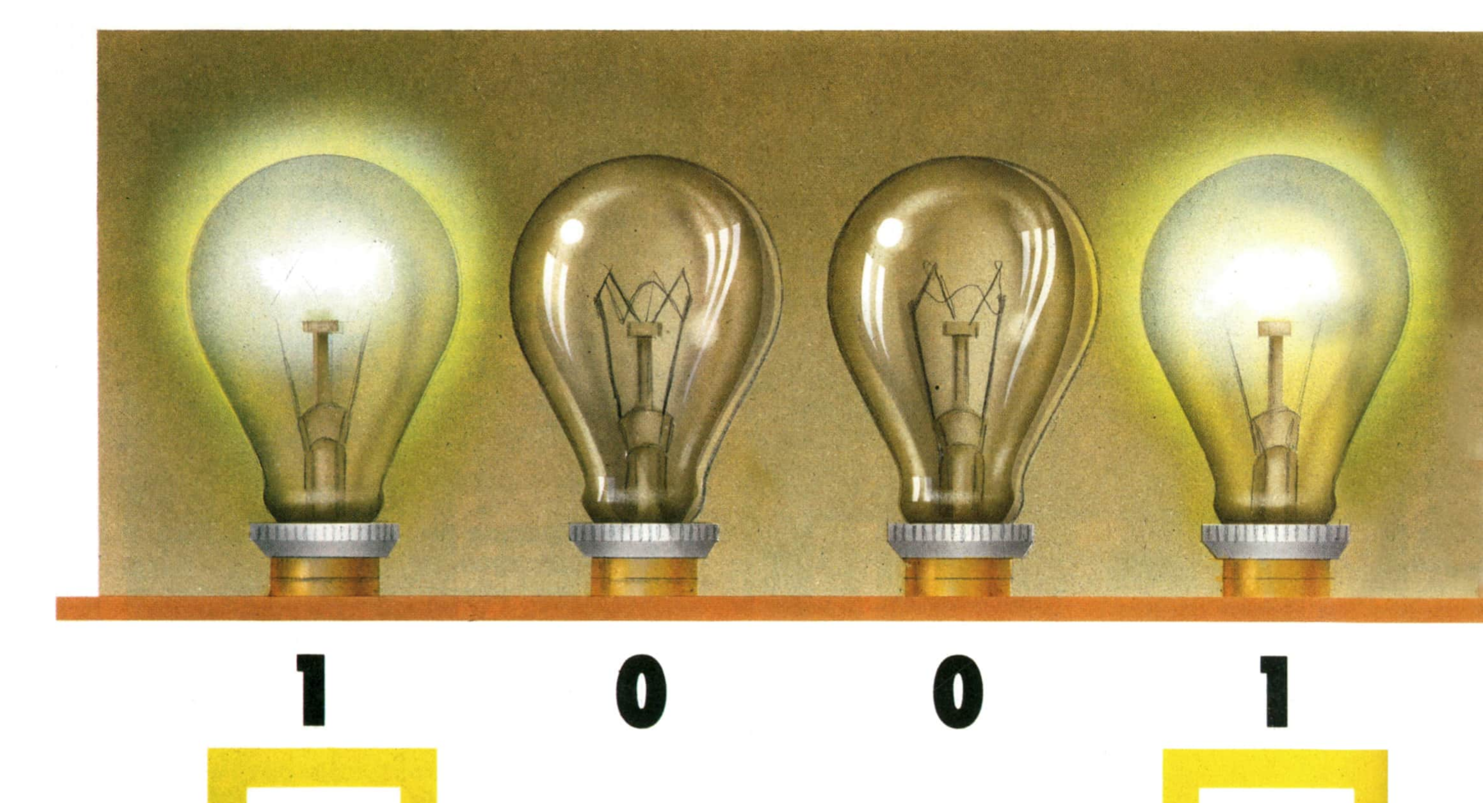
*A la izquierda, abajo: microprocesador, circuito integrado, conocido normalmente como* chip, *que es la unidad central de proceso en una microcomputadora y que ha hecho posible la asombrosa miniaturización de los ordenadores.*



**Características principales del MARK I**

| ***Características*** | ***Unidades*** |
|---|---|
| Longitud | 16 m |
| Altura | 2,2 m |
| Elementos móviles | 700.000 |
| Cables | 900 km |
| Conexiones eléctricas | 3.000.000 |
| Peso | 5.000 kg |

requería un complicado sistema de palancas. Al hablar de los contadores dijimos que con los biestables se podía diseñar un aparato que supiera contar. Con sólo uno que sabía contar hasta uno: (1 = encendido, 0 = apagado), con dos biestables el aparato contaba hasta tres (los dos apagados=0, el primero encendido y el segundo no=1, el segundo encendido y el primero apagado = 2, los dos encendidos = 3). Es evidente que si conectamos tres biestables, podremos contar más casos. Hasta siete. (Consulta de nuevo la figura **10**, el esquema de un contador.)




Fíjate que esto es así porque una bombilla sólo puede estar encendida o apagada. O sea que sólo disponemos de dos números diferentes. Nosotros tenemos diez cifras distintas. Es como si cada bombilla tuviera nueve colores además del apagado. Está claro que cuantos más colores tenga una bombilla menos bombillas harán falta para representar el mismo número.

### Las ventajas del sistema de numeración binario

Al hablar de los números, decíamos que el diez fue sin duda el primer sistema de numeración que se inventó. Es decir, diez cifras diferentes tantas como dedos de la mano. Pero no fue el único. Los Mayas, un pueblo que habitaba extensas áreas de Centroamérica, antes de que Colón pisara el suelo americano, contaban en base 20. Pero que una bombilla tenga diez colores es mucho más complicado que pedirle simplemente que se encienda y se apague. Por eso para hablar el lenguaje de las máquinas, hay que aprender a contar con sólo dos números, el cero y el uno, que es la mejor manera de representar la bombilla apagada y la bombilla encendida. Este sistema de numeración se llama numeración binaria, (*bi* significa «dos» en griego y bicicleta se llama así, porque tiene dos ruedas). Contar con sólo dos números es fácil. Mira el ejemplo.

| | | |
|---|---|---|
| 0 | 0 | Fíjate que no es difícil. Cuando cuentas con diez números, al |
| 1 | 1 | llegar al diez, pones un uno y un cero. Pues con dos al llegar |
| 2 | 10 | al dos haces lo mismo. |
| 3 | 11 | Contando en base diez (con diez números), al llegar al 99 |
| 4 | 100 | pones un uno delante de dos ceros y vuelves a empezar. En base |
| 5 | 101 | dos esto pasa al llegar al 11 (que es el tres). |
| 6 | 110 | De esta manera las bombillas encendidas y apagadas nos dibujan |
| 7 | 111 | un número. Desde luego hacen falta muchas más bombillas que |
| 8 | 1000 | cifras decimales para representar la misma cantidad. |

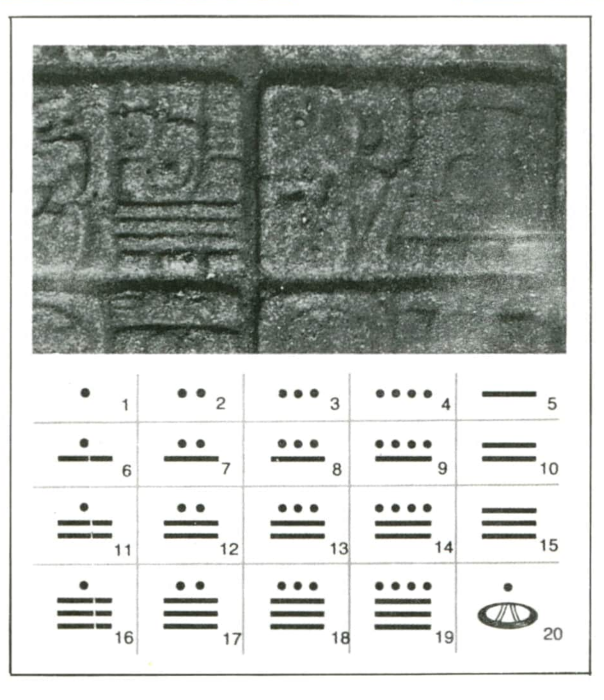
**1 0 0 1**

*La figura superior muestra la lógica del sistema de numeración binaria, que es el sistema que utilizan los ordenadores. Binario proviene de la raíz griega «bi», que significa dos, razón por la cual todas las cosas u objetos que tienen dos partes llevan siempre este prefijo: como bicicleta, por las dos ruedas. En el sistema de numeración binaria, el cero se asimila a la ausencia de corriente, mientras que el uno entra en acción cuando sí pasa. Operar en binario parece complicado, pero no lo es si te fijas en la tabla de la página anterior.*

Lo de las bombillas encendidas y apagadas es una manera de hablar. Lo que pasa en realidad es que llamaremos cero cuando no pasa corriente por un sitio y uno cuando sí pasa. Pero como sólo nos interesa saber si pasa o no pasa, no nos hace ninguna falta que circule mucha corriente.

## Operaciones en binario

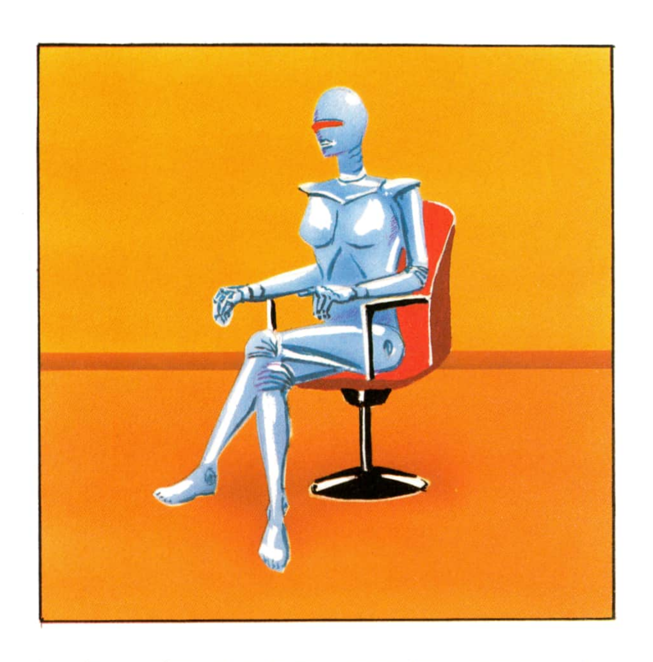
Sumar o restar, multiplicar o dividir en binario es lo mismo que en el sistema decimal, pero más sencillo. No hay que saberse las tablas de multiplicar, por ejemplo. Sólo hay que saber que uno por cero es cero y que uno por uno es uno. Que uno más nueve es diez y que cero más uno es uno.

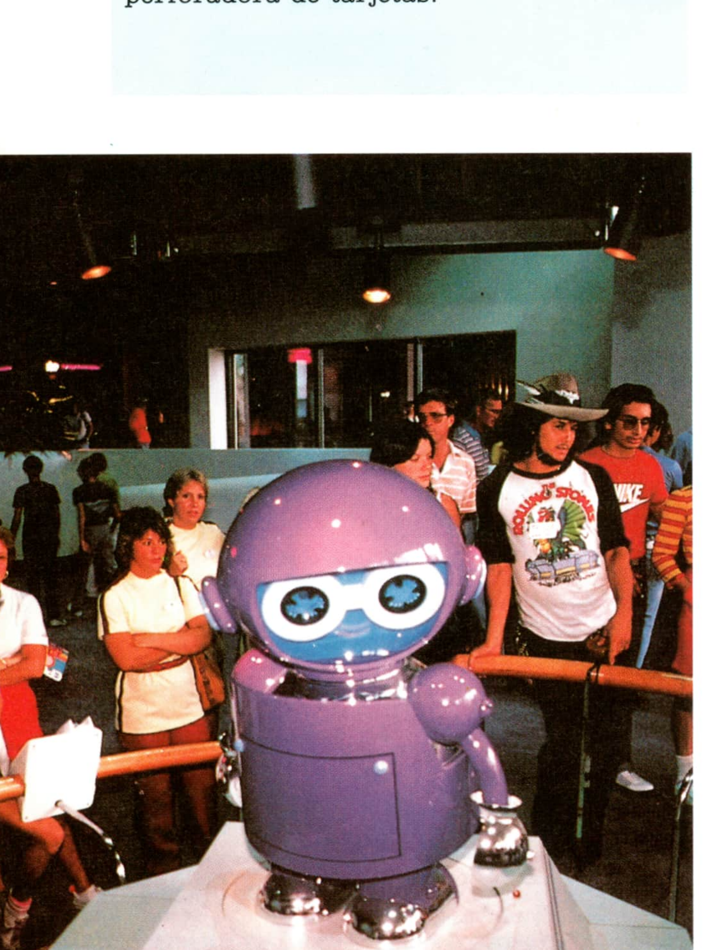



*Los Mayas, un pueblo que habitaba extensas áreas de Centroamérica antes de la llegada de los conquistadores españoles, desarrollaron un sistema de numeración vigesimal, es decir, en base 20, equivalente al número total de dedos de una persona (o sea de pies y manos). El sistema de numeración maya era tan avanzado que les permitió realizar numerosos cálculos astronómicos y de ingeniería.*

# LOS ORDENADORES

El aspecto externo puede variar, pero los principios básicos de funcionamiento de la mayoría de los ordenadores son los que siguen: se comienza por suministrar a la máquina instrucciones (programa) y datos a través de un dispositivo periférico de entrada. La memoria almacena toda la información y la unidad de control dirige el proceso, que se desarrolla en la unidad aritmética y lógica. La memoria registra los resultados y los hace legibles a través del dispositivo periférico de salida, que puede ser una pantalla, una impresora o una perforadora de tarjetas.





*A primera vista, parece que aprender a programar ha de ser algo extraordinariamente complicado. En realidad, construir un programa puede ser sencillo si partimos de la base de que la máquina exige para entendernos que seamos capaces de darle las instrucciones paso a paso, es decir, descomponiendo las operaciones y traduciéndolas al lenguaje máquina. Un sistema útil es jugar con Jane, el robot de Assimov que aparece en la ilustración superior. Izquierda, simpático robot que divierte a los niños.*

Como se dijo al principio, un ordenador o computadora es como un vídeojuego sencillo, pero que dispone de mucha memoria. Cuando lo compras no sabe hacer nada, excepto prepararse para aprender. Enseñarle a hacer lo que queremos es lo que en informática o computación llamamos programar. Una programador es algo parecido a un domador de animales, pero que tiene que vérselas con máquinas.
Para aprender a programar, un sistema excelente es jugar con Jane, el simpático robot intuitivo de los libros de Isaac Assimov, una de las personas que más se han esforzado en hacer de la robótica y el espacio una aventura maravillosa. Cada vez que aprendes a jugar a un juego nuevo, lo primero que debes hacer es saberte las reglas. El libro donde vienen las reglas de juego se llama *manual* en «informático», y tienes que acostumbrarte a manejarlo.

## SISTEMA OPERATIVO

### Manual de Jane

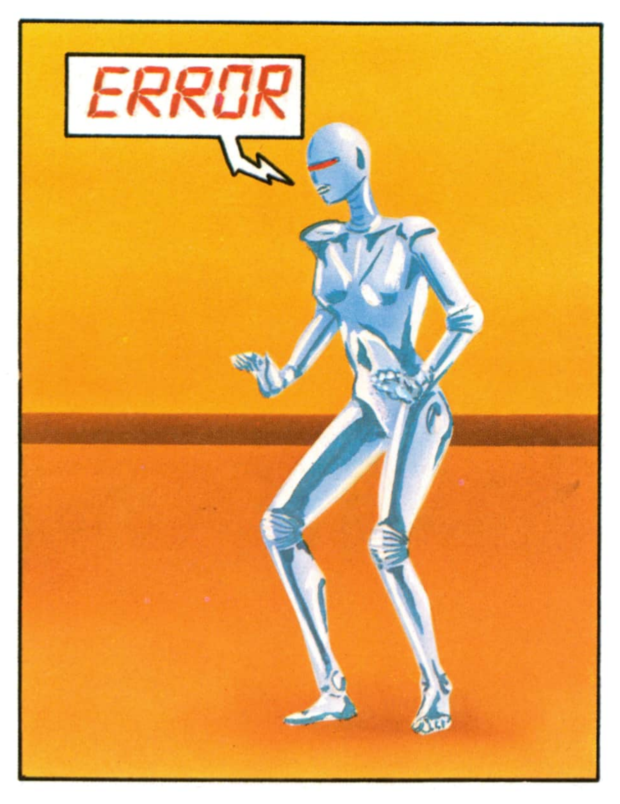
Jane es una mujer robot perfecta. Algo así como *Cortocircuito* FX-5, el increíble *Número 5* de la película. Si este libro fuera un manual de verdad seguirían unas largas y aburridas páginas sobre todo lo que puede hacer y cómo está hecho. Imagínate que a nivel físico puede hacer lo mismo que una persona y que su nivel mental le permite entender órdenes sencillas, como por ejemplo GIRA, AVANZA, etc., que sabe el nombre de las cosas habituales de una casa y que, si no lo sabe, lo puede aprender. Si le dices: «Jane, esto es una MESA». Y le señalas una mesa, ella aprende a distinguir una mesa. A todo lo que ya sabe le llamamos sistema operativo.

*Cómo se le da una orden*

El manual también te indica cómo debes darle cada orden para que la entienda. Por ejemplo, para que gire, tienes que decirle: «GIRA 90 DERECHA». El manual lo escribe así: GIRA / n / / DIR /

Significa que la orden GIRA debe ir acompañada de un número (los grados) y una dirección (derecha o izquierda). Si le das la orden incompleta o equivocada: «GIRA 30 ARRIBA», Jane responderá «ERROR». Como leerse un manual es muy aburrido, supondremos que ya conocemos su contenido y te lo explicaremos a medida que haga falta.



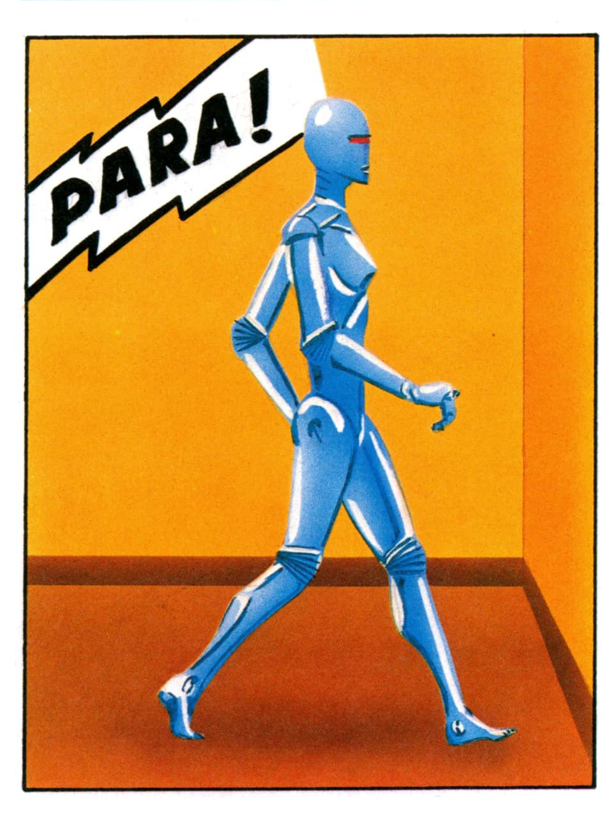



*Jane ha obedecido nuestra orden y se ha puesto en marcha. Pero, al estar equivocada, responde, gracias a su capacidad de discernimiento del sistema operativo, «ERROR». Periféricos de entrada de datos, memoria y periféricos de salida son las partes básicas del ordenador.*

*Enseñemos a Jane a ir a la cocina*

Si a Jane le decimos de entrada «Ve a la cocina», se quedará parada sin hacer nada porque no entiende la orden. Para que nos obedezca, le tenemos que ir diciendo paso a paso todo lo que tiene que hacer para cumplir nuestro mandato inicial de «Ve a la cocina». Es decir, le enseñaremos todos los actos que debe hacer para ir a la cocina, y le diremos que se acuerde siempre que cuando oiga esta orden debe ejecutar el programa «Ve a la cocina». Cualquiera de nuestras decisiones se compone de un conjunto de pasos que hemos de descomponer ordenadamente para que el robot o la computadora puedan entenderlos. Acuérdate de lo que hace él mimo en el teatro para escribir en un papel. La de movimientos que utiliza para agarrar el bolígrafo. Guiándole paso a paso, conseguiremos que Jane vaya a la cocina. Más tarde, podremos indicarle que agarre un vaso con cuidado, etcétera.

«Jane, GIRA 45 DERECHA» (hacemos que quede encarada hacia la puerta).
«Jane, AVANZA 3» (que avance tres pasos y se plante en el pasillo).
«Jane, GIRA 45 DERECHA» (la encaminamos hacia la cocina).
«Jane, AVANZA 3» (ya está en la cocina).




*Mientras probamos nuestro programa «Jane ve a la cocina» «Jane gira y avanza», descubrimos que la puerta está cerrada. Antes de que nuestra simpática Jane se rompa las narices, le gritamos PARA, y procedemos a rectificar el error.*

*Vamos a programar*

Programar consiste en decirle por adelantado al robot todo lo que tiene que hacer cuando oiga que le ordenamos «Jane, ve a la cocina».

Para hacer este programa, basta con que le digamos todo lo que tiene que hacer. Pero como no estaremos para orientarle, tenemos que prever todos los casos posibles. Empecemos por ver lo que pasa con la primera instrucción o comando «GIRA 45 DERECHA». Para que esta orden sirva en todos los casos, Jane 5 debe estar siempre en el mismo sitio. Podemos conseguirlo si le ordenamos que vuelva siempre a su sitio cuando acabe cada trabajo. Con esta condición, la primera instrucción funcionará siempre.

*Probemos nuestro programa*

«Jane ve a la cocina» «Jane gira y avanza tres pasos».

¡Cielos! ¡La puerta que da al pasillo está cerrada! «PARA», le gritamos antes de que rompa la puerta o la puerta le rompa a ella. «PARA» es una orden que entiende. Hemos cometido un error en nuestro programa. No será el último. Nos hemos olvidado de la puerta. No hay más remedio entonces que retocar el programa. Debemos descomponer la instrucción «AVANZA 3», porque hay que hacerle abrir la puerta.

AVANZA 2
ABRE PUERTA

Si «Jane» no sabe lo que significa ABRE, se lo podemos explicar diciéndole que se trata de: agarrar picaporte, mover mano hacia abajo, extender el brazo hacia adelante (o flexionarlo hacia atrás si la puerta se abre hacia dentro), soltar picaporte. Abrir es, en realidad, un conjunto de instrucciones, es decir, un programa. Pero lo podemos considerar como una nueva palabra en el diccionario de Jane.

AVANZA 1

*Volvamos a probar*

Cuando hagas un programa, recuerda siempre que debes preguntarte cómo y cuándo se puede romper. Así que pensaremos: «¿Qué pasará si la puerta está abierta?» Pues lo probamos. Dejamos la puerta abierta y decimos: «Jane, ve a la cocina».

Jane avanza hasta la puerta y se para buscando el picaporte. «PARA», le ordenamos.

Si la puerta puede estar tanto abierta como cerrada, no hay más remedio que indicarle lo que tiene que hacer en cada caso y, lo que es más importante, preguntárselo a ella. Según el manual, se le tiene que decir:

SI / condición / ENTONCES / acción /

De modo que esta parte del programa queda así:

```
AVANZA 2
SI PUERTA NO ABIERTA, ENTONCES
ABRE PUERTA
AVANZA 1
```

Esta parte del programa ya es correcta. Fíjate que hemos tenido que sustituir la acción de salir al pasillo «AVANZA 3» por tres instrucciones (además de todas las que necesitamos para hacerle abrir la puerta). Siempre pasa lo mismo: todo se complica.

## Instrucciones repetitivas

La primera instrucción, «GIRA 45 DERECHA», plantea el inconveniente de que Jane tiene que estar siempre en la misma posición inicial. Puesto que un programa es tanto mejor cuantas menos condiciones haya que ponerle, vamos a hacer que Jane se encare ella sola hacia la puerta. Para ello contamos con la instrucción:

VE A / número de instrucción /

porque evidentemente Jane se acuerda del número de orden de cada instrucción. Así que le diremos:

```
1 Si no ves la puerta,
ENTONCES GIRA 1 DERECHA
VE A 1
```

De esta manera, Jane girará hacia la derecha en tanto no vea la puerta. Dado que es más fácil decirle: «MIENTRAS no veas puerta, HAZ GIRA 1 DERECHA», le enseñaremos la instrucción:

MIENTRAS / condición / HAZ / instrucción /

Para encarar a Jane con la puerta, también podríamos haberle dicho:

```
1 GIRA 1 DERECHA
SI no ves la puerta, ENTONCES VE A 1
```

De este modo, nuestra Jane giraría a la derecha hasta que viera la puerta. Así que inventaremos la instrucción:

REPITE / instrucción / HASTA / condición /

*Haz*
Las instrucciones repetitivas, es decir, aquellas que repiten una instrucción o una serie de instrucciones un número de veces, son una de las bases de la programación. Existe todavía una tercera forma de indicar una instrucción repetitiva, que se emplea cuando se sabe cuántas veces debe repetirse una acción.

HAZ / número veces / / instrucción /

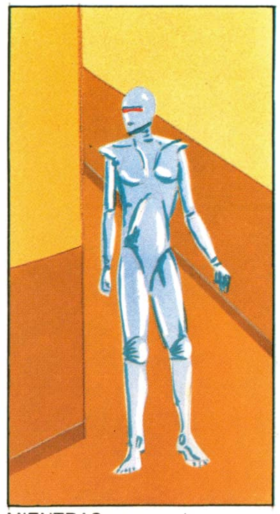
Con la introducción del MIENTRAS, el programa «VE a la cocina» queda así:

```
MIENTRAS no veas la puerta,
   HAZ GIRA 1 DERECHA
MIENTRAS no llegues a la puerta,
   HAZ AVANZA 1
SI la puerta no está abierta,
   ENTONCES ABRE PUERTA
AVANZA 1
GIRA 45 DERECHA
AVANZA 3
SI puerta no abierta,
   ENTONCES ABRE PUERTA
AVANZA 1
```

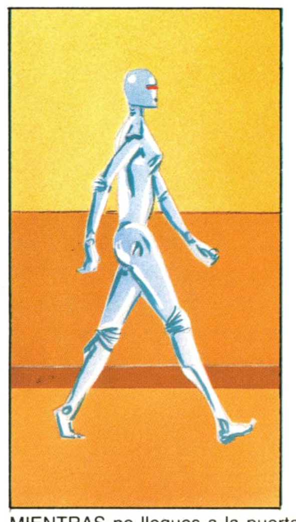
*(Véase el dibujo de la página 36).*

## Rutinas

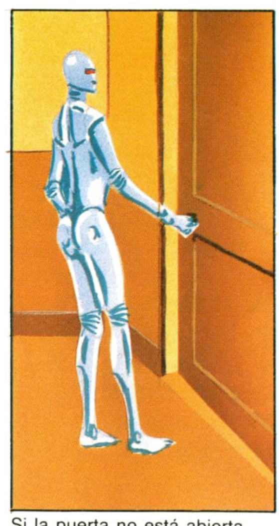
La informática o computación es el arte de escribir las cosas sólo una vez. No lo olvides nunca. Para ir educando a Jane, tenemos que aprovechar todo lo que le hemos enseñado. Cada vez que aprende algo nuevo, le ponemos un nombre. Lo mismo que hemos hecho cuando le hemos enseñado a abrir puertas. A cada grupo de instrucciones que por sí solas hacen algo le llamaremos subprograma o subrutina, o procedimiento o rutina. De este modo, podemos decirle que las cuatro primeras instrucciones forman la rutina «Sal al pasillo». Nuestro programa queda ahora con las instrucciones de la página siguiente.



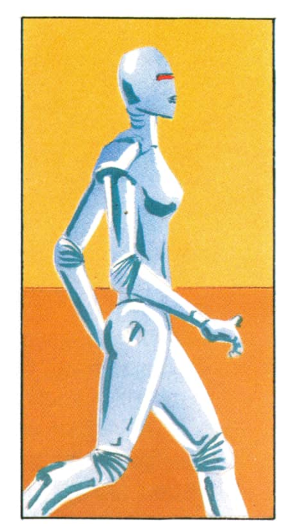
MIENTRAS no veas la puerta,
HAZ GIRA 1 DERECHA



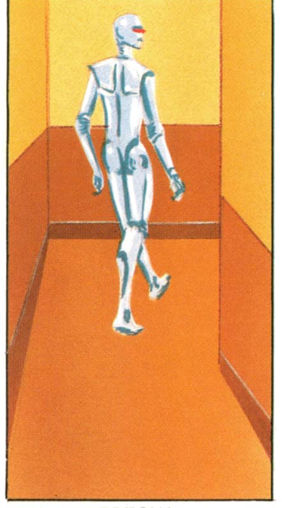
MIENTRAS no llegues a la puerta,
HAZ AVANZA 1



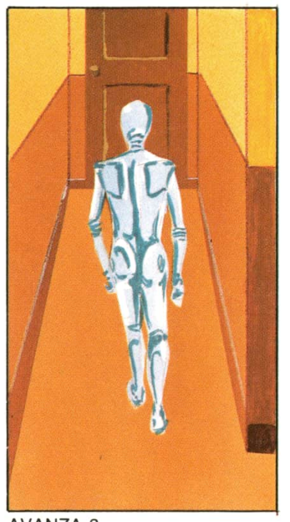
Si la puerta no está abierta,
ENTONCES ABRE PUERTA



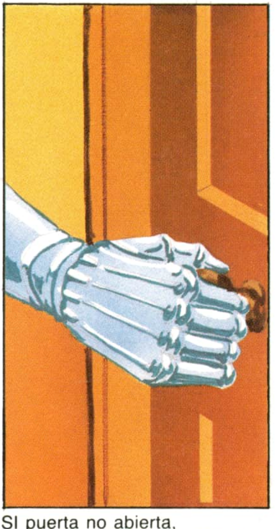
AVANZA 1



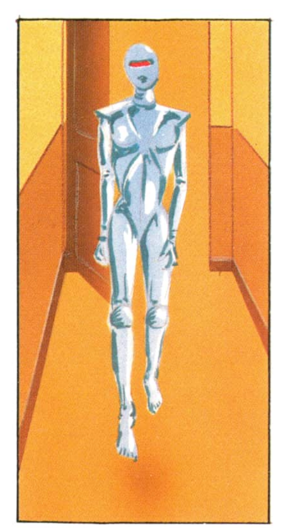
GIRA 45 DERECHA



AVANZA 3



SI puerta no abierta,
ENTONCES ABRE PUERTA



AVANZA 1

*La ilustración muestra todos los pasos que ordenamos a Jane que cumpla para que realice el programa «VE a la cocina». Aquí se han utilizado la condición MIENTRAS y una serie de instrucciones repetitivas.*

```
SAL AL PASILLO

MIENTRAS no llegues a la puerta de la cocina, HAZ AVANZA 1
SI la puerta no está abierta,  ENTONCES ABRE PUERTA
AVANZA 1
```

*Acabemos el programa*

Si ahora queremos que Jane nos traiga un vaso de agua, el programa empezará directamente: «Ve a la cocina». Luego le tendremos que hacer buscar el vaso, agarrarlo, buscar el grifo del agua fría, abrirlo, llenar el vaso, cerrar el grifo. La última instrucción sería: «Ve al comedor», que es la rutina inversa de ir a la cocina y que, naturalmente, también le tendríamos que decir cómo se hace.

Como ves, un programa está hecho de muchos subprogramas o rutinas.

Programar no es más que dividir un proceso en partes y cada una de éstas en más trozos hasta llegar a tener sólo instrucciones sencillas. Que a su vez se acabarán traduciendo en ceros y unos, que es lo único que entienden las máquinas.

## LA MEMORIA DE LA COMPUTADORA

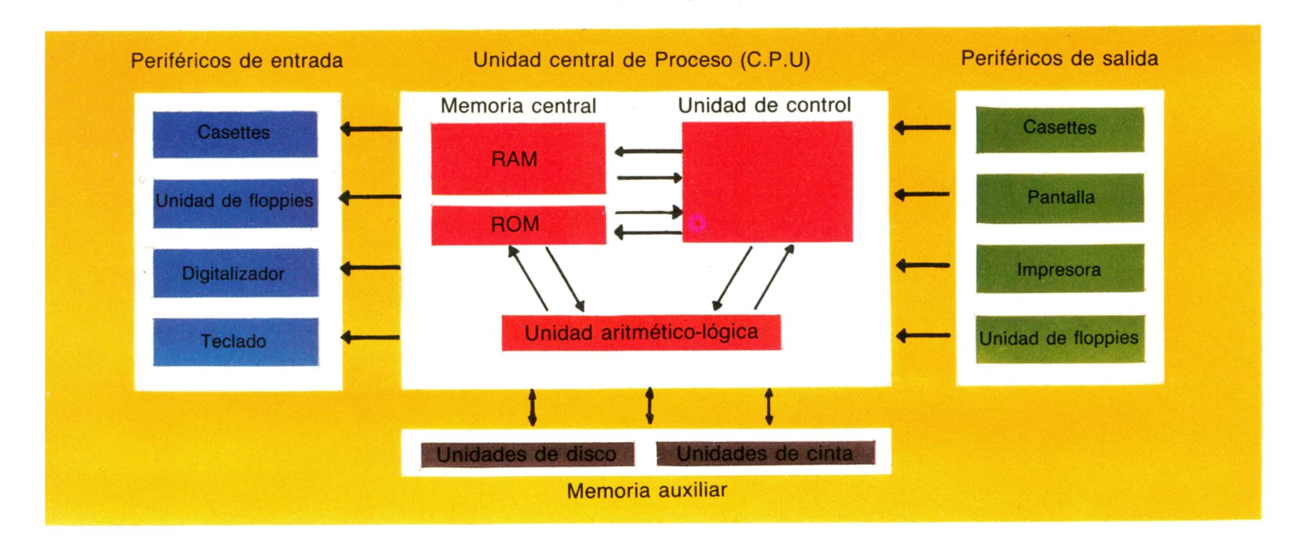
Empecemos por el principio. Jane sólo tiene memoria. Puede guardar muchos ceros y unos, muchas bombillas encendidas o apagadas. Pero lo único que sabe es que las bombillas van agrupadas. Digamos que de 16 en 16. Es el momento de hablar en «informático». Diremos *bit* cuando pensemos en bombillas. Un bit a cero es la bombilla apagada y a uno si está encendida. Es la unidad de información. A los grupos de 16 bit los llamaremos palabras. Como si el bit fuera una letra, vaya. Las palabras están guardadas ordenadamente dentro de la máquina como si fueran casas y cada una tiene su dirección.




*Arriba, los elementos que forman el hardware de una computadora pueden estar integrados en varios cuerpos o disponerse de manera separada. Aquí aparecen separados, pero formando parte del sistema global que es la computadora. Tenemos en el dibujo periféricos de entrada, el teclado, y de salida, la pantalla o monitor, la impresora, así como periféricos de entrada-salida, el cassette o cinta magnética. Abajo, esquema de la Unidad Central de Proceso o C.P.U* (Central Processing Unit), *que es la parte principal de un ordenador, pues controla y realiza todas las operaciones lógicas y los cálculos numéricos e interpreta las instrucciones y ordena su ejecución.*

*Dos clases de memoria*

En la memoria, distinguiremos dos partes. En una se pueden guardar datos (combinaciones de 0 y 1) y se pueden leer. La llamamos RAM (del inglés Random Acces Memory) que, en inglés, significa memoria de acceso aleatorio, es decir, que se puede acceder a cualquier

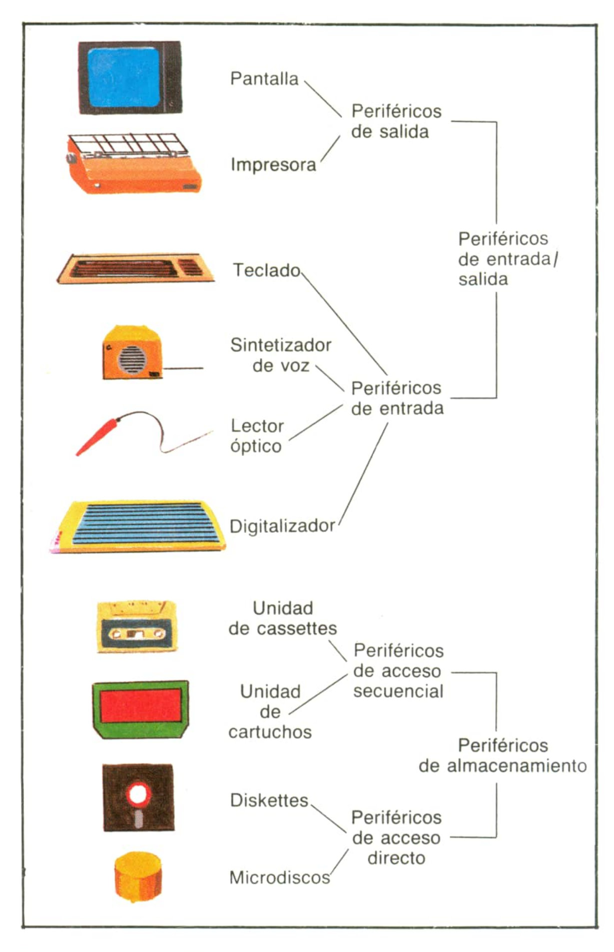


dirección ya sea para guardar o para leer. La otra parte sólo se puede leer, no permite que se almacene nada en ella. Recibe el nombre de ROM (Read Only Memory, o sea, Lectura Sólo Memoria). No se puede borrar, ni siquiera cuando se apaga la máquina. Al conectar el ordenador, ejecuta los programas que están en la ROM, con lo que puede empezar a funcionar. Tiene también unos circuitos que saben sumar, restar y decidir si un número es mayor que otro y si un bit está en cero o en uno. Estos circuitos constituyen lo que se denomina Unidad Aritmético - Lógica (UAL).

*El procesador*
Cuenta asimismo con un procesador, que es un aparato que conecta y desconecta circuitos, en función de las señales que recibe. Sabe ir a buscar una determinada palabra, si le indicamos en qué dirección está, y sabe interpretarla.
El conjunto de la memoria, la UAL y el procesador recibe el nombre de CPU (Unidad Central de Procesos; en inglés *Central Processing Unit).*

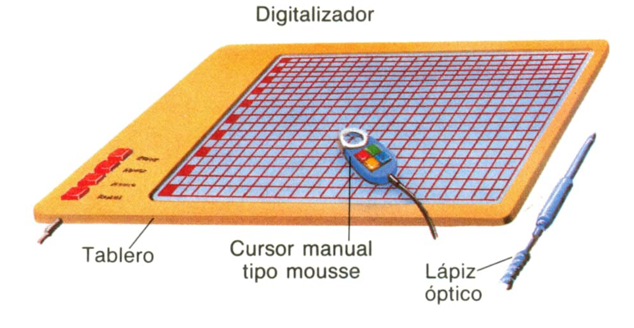
## PERIFÉRICOS

Se llaman periféricos porque están en la periferia del esquema de un ordenador (recuerda que la periferia es la parte exterior de un objeto, cosa o figura). Constituyen el medio de comunicación entre nosotros y la máquina. Por los periféricos de entrada le proporcionamos información a la máquina, mientras que por los de salida nos devuelve aquélla debidamente procesada.
El teclado es el periférico de entrada más utilizado en la actualidad, aunque el «ratón», o *mouse* parece que va imponiéndose en los ordenadores personales y microordenadores.
El ratón es un dispositivo pequeño y juguetón que se maneja con la mano. Conectado al teclado de la computadora en unos casos o a un periférico de salida en otros, permite mover el cursor (raya o marca luminosa que indica el lugar que ocupará el símbolo que quiere visualizarse en pantalla) a lo largo y ancho de la pantalla desplazándolo encima de una mesa o superficie plana.




*Principales periféricos de un ordenador.*

—*Lectores ópticos:* Lápices capaces de leer el código de barras, cada día más empleados en los supermercados.
— *Digitalizadores:* Aparatos unidos por un cable a la máquina, que siempre saben dónde están. Un digitalizador muy




*Arriba, dibujo de un digitalizador, periférico de entrada que permite convertir informaciones gráficas o dibujos en informaciones numéricas para que las entienda el ordenador. Funciona con cursor manual tipo mousse o con lápiz óptico.*

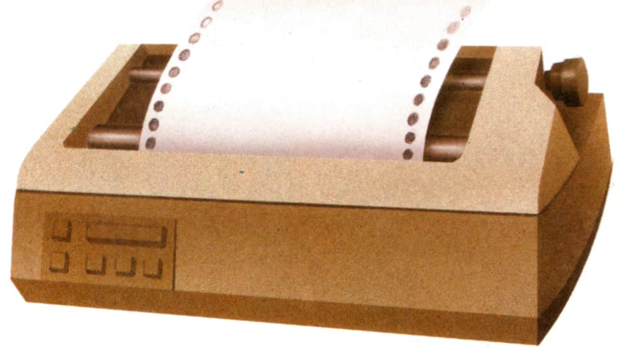
común es el «lápiz óptico» que, como su nombre indica, es un lápiz provisto de una célula fotoeléctrica que hace que la computadora reconozca los símbolos o caracteres cuando el lápiz los puntea.

*El lápiz es el tipo de digitalizador más corriente utilizado hasta hoy. Permite almacenar dibujos o gráficos en la memoria del ordenador repasando su contorno con el lápiz en el tablero, que irá transmitiendo punto a punto las características del dibujo, tal como muestra la fotografía superior.*

No hace muchos años (todavía se utilizaban hace una década), la única entrada eran las lectoras de tarjetas perforadas. En la actualidad, se está investigando el modo de que un micrófono pueda leer nuestra voz. Seguramente habrás visto llaveros que pitan si les silbas, indicándote dónde se encuentran. Te entienden y te contestan.

## Periféricos de salida

Los más empleados son la pantalla y la impresora. La primera es más rápida, pero la segunda escribe en un papel. La *pantalla* es aquella parte del ordenador donde visualizamos los datos, las instrucciones o comandos que damos a la máquina mediante el teclado, el ratón o el lápiz óptico o que el mismo ordenador genera gracias a la información que tiene siempre almacenada, la ROM.
Las *impresoras* son unos aparatos que, como su nombre indica, sirven para que la información que nos ha elaborado la computadora aparezca escrita en un papel. De este modo podremos utilizarla en cualquier sitio sin necesidad de consultarla en la pantalla. Existen muchas clases de impresoras (por puntos, por margarita, por chorro de tinta, por láser, etc.), pero todas nos muestran los resultados del procesamiento sobre un papel.



*Impresora de matriz de puntos, periférico de salida que sirve para que la información que nos ha elaborado la computadora aparezca escrita en papel.*



Plotter de mesa

*Arriba, dibujo de un plotter de mesa, periférico de salida que efectúa dibujos automáticamente mediante el desplazamiento de un útil, plumilla, bolígrafo presurizado o rotulador sobre una superficie (papel, transparencia, etc.,) que es el soporte. En unos plotters es el soporte lo que se mueve, mientras que en otros es el útil del dibujo lo que se desplaza y el soporte permanece fijo. Gracias a los plotters se pueden realizar dibujos muy complejos. Abajo, el dibujo muestra la relación existente entre los distintos dispositivos para el almacenamiento de memoria.*

Sin embargo, pantalla e impresora tampoco son las únicas:

- *Plotters:* máquinas que saben dibujar. Poseen varios rotuladores que pueden subir y bajar y moverse en una dirección. En la otra se mueve el papel.
- *Sonido:* No sólo es posible en los llaveros, sino que también en coches que hablan. Pero lo que dicen se ha tenido que grabar antes.

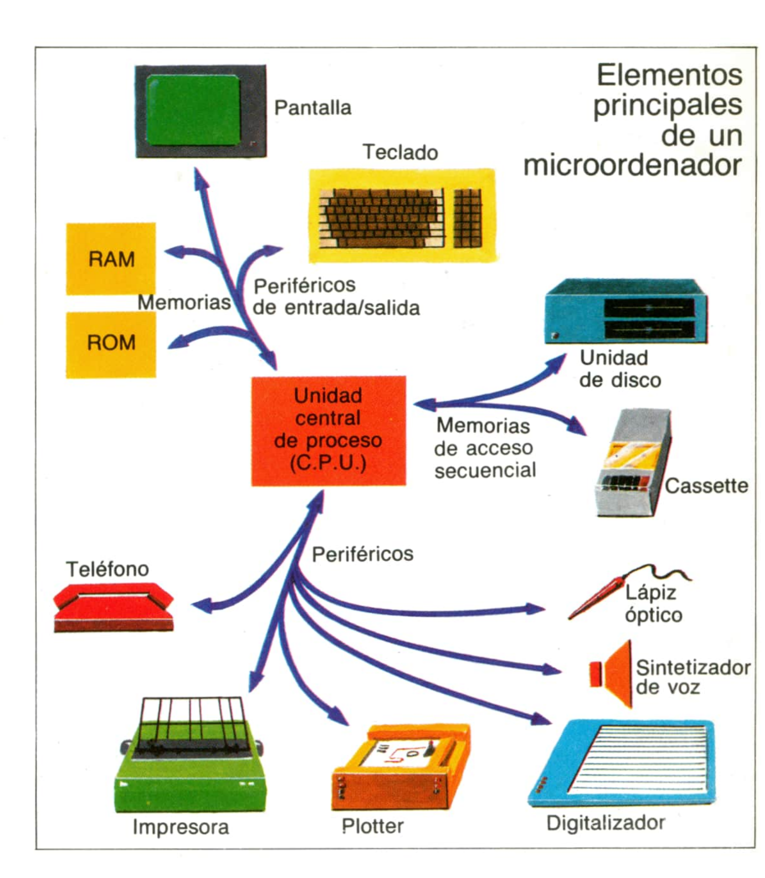
## Periféricos de entrada y salida

Cuando apagas el ordenador, toda la memoria RAM se borra y la ROM, como ya se explicó antes, no. Así que todo lo que hayas hecho se pierde. Para evitarlo, existe la memoria exterior, es decir, lugares donde la memoria no se pierde porque se graba igual que el sonido. Se trata de los cassettes, los discos y los diskettes o *floppies*. Son de entrada y salida porque sirven para entrar datos o programas o para guardarlos.

Comparación entre distintos soportes de información

## CÓMO FUNCIONA UN ORDENADOR

Si vieras el interior de un ordenador, comprobarías que está lleno de chips, esa especie de arañita de color negro. Si te fijaras bien, verías que hay uno más grande: es el microprocesador. Los otros, más pequeños, constituyen la memoria. Verías también unas rayas doradas que van por todas partes: son los *buses* o autobuses, por los que circula la información en forma de ceros y unos. El microprocesador se encarga de ir cada vez a una u otra dirección de memoria y hacer lo que allí se encuentra. El orden de actuación del microprocesador viene dado por los programas.




*A la derecha, el dibujo muestra los principales elementos que pueden formar parte del equipo informático acoplado a un microordenador. Si bien la unidad central y la capacidad de memoria son las partes esenciales, no hay que olvidar su capacidad para incorporar todo tipo de periféricos que permitan una amplia gama de trabajos. Ciertos microordenadores incorporan en un mismo soporte las distintas unidades que en el dibujo aparecen como independientes.*

### Esquema de funcionamiento del ordenador

Memoria

Entrada de datos

Unidad de control

Unidad aritmético-lógica

Salida de datos

*El dibujo muestra el esquema de funcionamiento de un ordenador, cuya estructura interna realiza la función del tratamiento y conservación de la información, mientras que los periféricos que se le acoplan realizan dos etapas más, que anteceden y suceden a la etapa central. Es precisamente a través de los periféricos como se producen la entrada y la salida de información.*

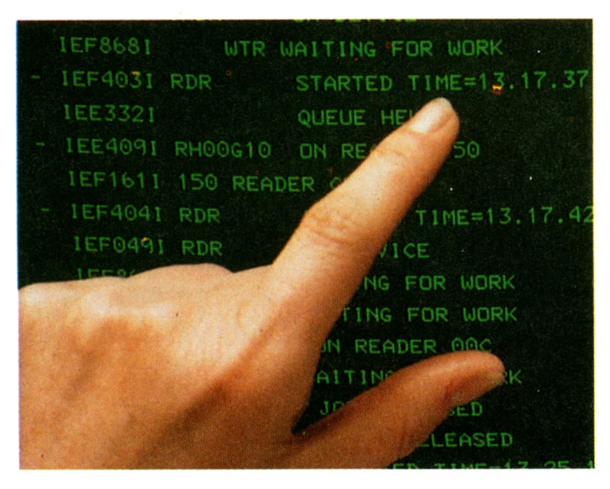
# EL LENGUAJE DE LAS MÁQUINAS

Como hemos supuesto que el procesador leerá los bits de 16 en 16, cada instrucción de un programa debe caber en 16 bits. Son $2^{16}$ posibilidades las que se pueden lograr con 16 bits.
Así que convendremos, por ejemplo, que los cuatro primeros bits le dirán qué instrucción debe hacer. Con 4 bits podemos hacer 16 combinaciones diferentes. Tenemos pues un idioma de 16 palabras. Se trata entonces de saber cuáles son las palabras base que nos permitirán construir todo el lenguaje. Los otros 12 bits servirán para concretar las instrucciones. Pero dado que para empezar a hablar el lenguaje de las máquinas hay que saber un poco más sobre cómo están hechas, explicaremos su funcionamiento, empezando por una muy fácil.

## La máquina de Arturo

La máquina de Arturo es una computadora muy sencillita. Tanto que podrás entender muy bien cómo funciona. Sin embargo, posee el mínimo necesario para poder realizar programas, aunque muy sencillos. La memoria RAM está dividida en dos partes: una operativa y la otra sólo de almacenamiento. En la parte operativa, muy pequeña, se pueden hacer los cálculos y las entradas y las salidas. Tiene una sola salida y una sola entrada. Es decir que sólo podrá sacar o entrar cada vez una palabra de 16 bits. A estas partes las llamaremos *registro de entrada* y *registro de salida*. Registro, del inglés *record*, es la palabra informática que designa una unidad lógica de información.
El cálculo se efectúa en dos registros, llamados *acumulador A* y *acumulador B*. Reciben estos nombres porque lo único que pueden hacer en cuanto al cálculo es acumular información (sumar o restar). Para ello es necesario poner en estos registros lo que esté en una determinada dirección de memoria, sumarlo y restarlo de lo que tengan o llevar información de los acumuladores a las posiciones.
Un microprocesador necesita dos registros más para poder actuar. Uno le sirve para saber en qué dirección de memoria está y el otro para saber en qué instrucción del programa se encuentra.

*Instrucciones de un programa en pantalla.*



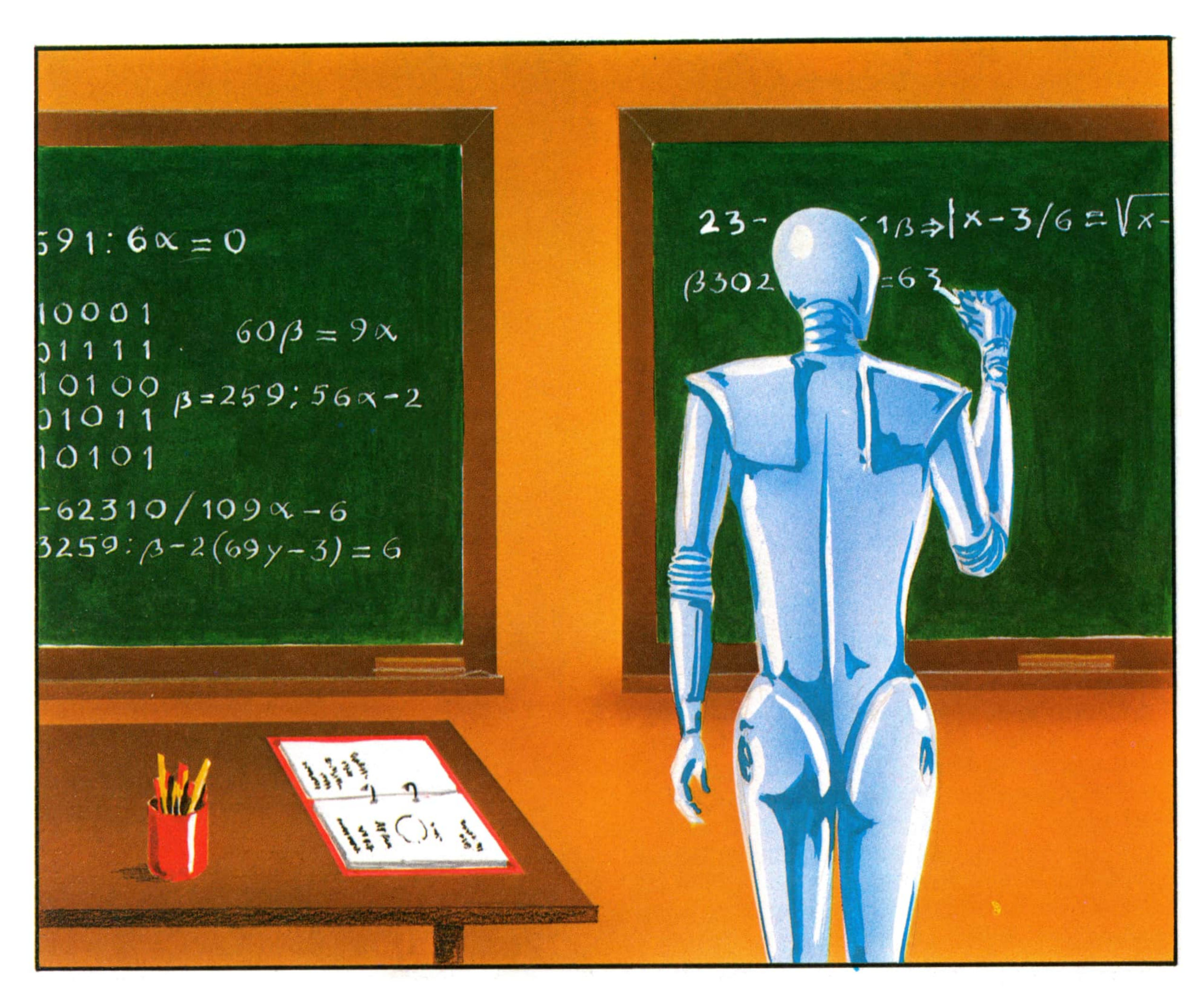

## ¿Cómo serán las instrucciones del programa?

Puesto que, tal como hemos dicho antes, la instrucción propiamente dicha ocupa 4 bits, nos quedan todavía 12 para concretar la instrucción. Una de las primeras cuestiones que debemos establecer es si trabajaremos con el acumulador A o el B. Dado que sólo se trata de dos casos, necesitaremos un único bit (01), que será el quinto.
Otra cuestión muy importante es decirle a qué dirección de memoria debe ir.
Como puede haber muchas, utilizaremos los 9 últimos bits.
Todavía nos quedan dos bits que usaremos para decirle al microprocesador si lo que viene detrás es una dirección de memoria o el valor absoluto de un número con el que tiene que hacer algo.

## El lenguaje de la máquina de Arturo

Supongamos que queremos hacer un programa para que la máquina de Arturo actúe como una sumadora. Siempre que hagas un programa, tienes que empezar por pensar cómo se lo dirías a Jane. Así, por ejemplo:

```
PREGUNTA un número
PREGUNTA otro número
SÚMALOS
ENSEÑA suma
PARA
```




*Jane está desarrollando un programa en lenguaje máquina. Para ello hemos decidido que los dos acumuladores o registros A y B son las pizarras donde apunta los resultados de los cálculos, mientras que la memoria es el bloc con las páginas numeradas que tiene encima de la mesa. El programa de esta página es el resultado.*

Ésta es la manera más sencilla; pero tiene la pega de que si se lo decimos así a Jane, ésta se puede liar con los números. Para evitarlo, le tenemos que decir dónde debe guardar los números. Si suponemos que los acumuladores A y B son dos pizarras donde apunta los resultados de los cálculos y que la memoria es un bloc con las páginas numeradas, le podemos decir:

*Programar en lenguaje máquina*

De este modo, le puedes indicar cómo hacer todas las operaciones que quieras, siempre que tú no te líes con las direcciones de memoria, es decir, con las páginas de la libreta donde debe ir apuntando los cálculos intermedios.

Y en la misma libreta, pero a partir de la primera página, es donde le dices que guarde el programa. Por ello la primera dirección de memoria que hemos utilizado ha sido la 100 y por eso en la máquina de Arturo tendremos que numerar las instrucciones, para que sepa dónde debe guardar cada una. Por esa razón, tiene un

```
PREGUNTA un número y ponlo en la pizarra A
PREGUNTA un número y ponlo en la pizarra B
PON el número de A en la página 100
SUMA el número de la página 100 y el número de la pizarra A; escribe el resultado en B
ENSEÑA lo que tienes en B
```

contador de instrucciones del programa, para saber siempre lo que tienes que hacer y por ese mismo motivo, la última instrucción tiene que ser siempre PARA.

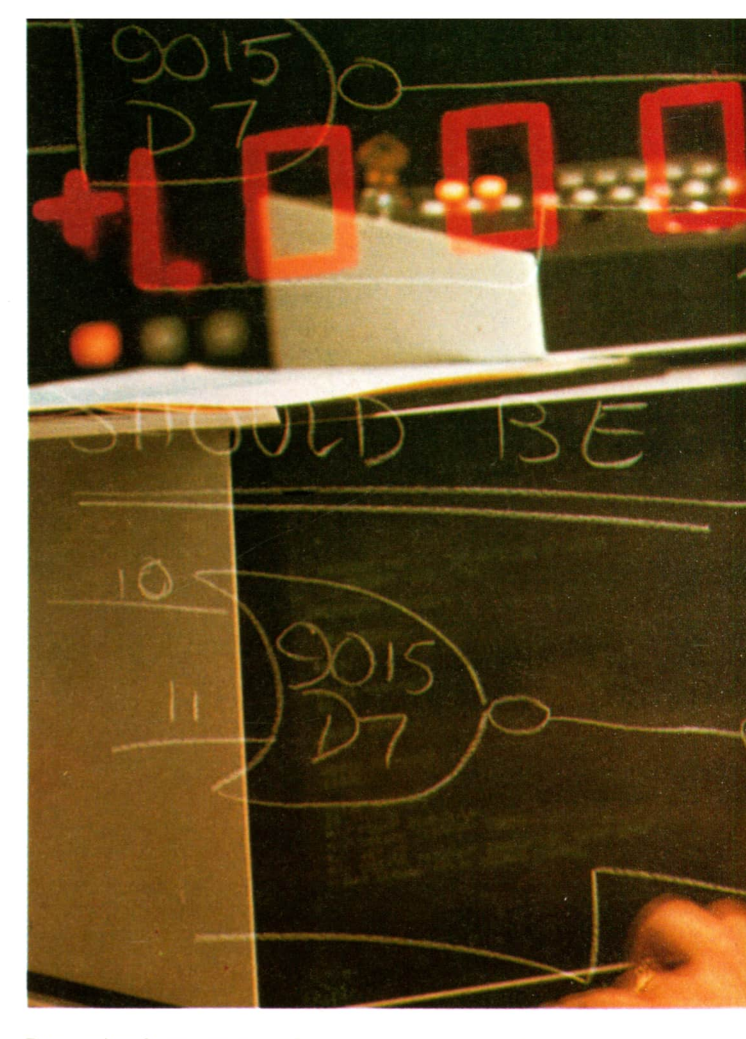
### Vocabulario de Arturo

Para efectuar este sencillo programa hemos necesitado las siguientes instrucciones: PREGUNTA, PON, SUMA, ENSEÑA, PARA. Puesto que, tal como hemos dicho, Arturo entiende 16 palabras, todavía nos quedan 11 más por definir. Lo haremos más adelante. De momento, vamos a escribir este programa en el lenguaje de Arturo.

0 METE en el acumulador A el número que entremos en el registro de entrada
1 METE en el acumulador B el número que entremos en el registro de entrada
2 GUARDA el número del acumulador A en la dirección 100
3 SUMA lo de la dirección 100 con lo del acumulador B y PON el resultado en el acumulador B
4 ENSEÑA el valor del acumulador B
5 PARA

Que es lo mismo que decir:

0 METE A (introduce un valor en el acumulador A)
1 METE B (introduce un valor en el acumulador B)
2 GUARDA A en 100 (guarda el valor de A en la dirección 100)
3 SUMA en B de 100 (suma al valor de B el de 100 y ponlo en B)
4 SACA B (enseña el valor de B)
5 PARA




*Para evitar la programación con ceros y unos, lo que resulta, además de aburrido, laborioso, se inventaron los ensambladores o lenguajes de bajo nivel, que no son sino lenguajes muy sencillos de traducción inmediata al código binario. Han sido de gran utilidad para construir sencillos programas adecuados para personas no profesionales de la programación.*

Ahora sólo nos falta escribirlo con ceros y unos. Según el código de Arturo, que ya conocerás más adelante, sería así:

| Línea | Instrucción | Acumulador | Forma | Valor | |
|---|---|---|---|---|---|
| 0000 | 1101 | 0 | 00 | 000000000 | |
| 0001 | 1101 | 1 | 00 | 000000000 | |
| 0010 | 0001 | 0 | 01 | 001100100 | (100 en binario) |
| 0011 | 0010 | 1 | 01 | 001100100 | |
| 0100 | 1110 | 1 | 00 | 000000000 | |
| 0101 | 1111 | 0 | 00 | 000000000 | |

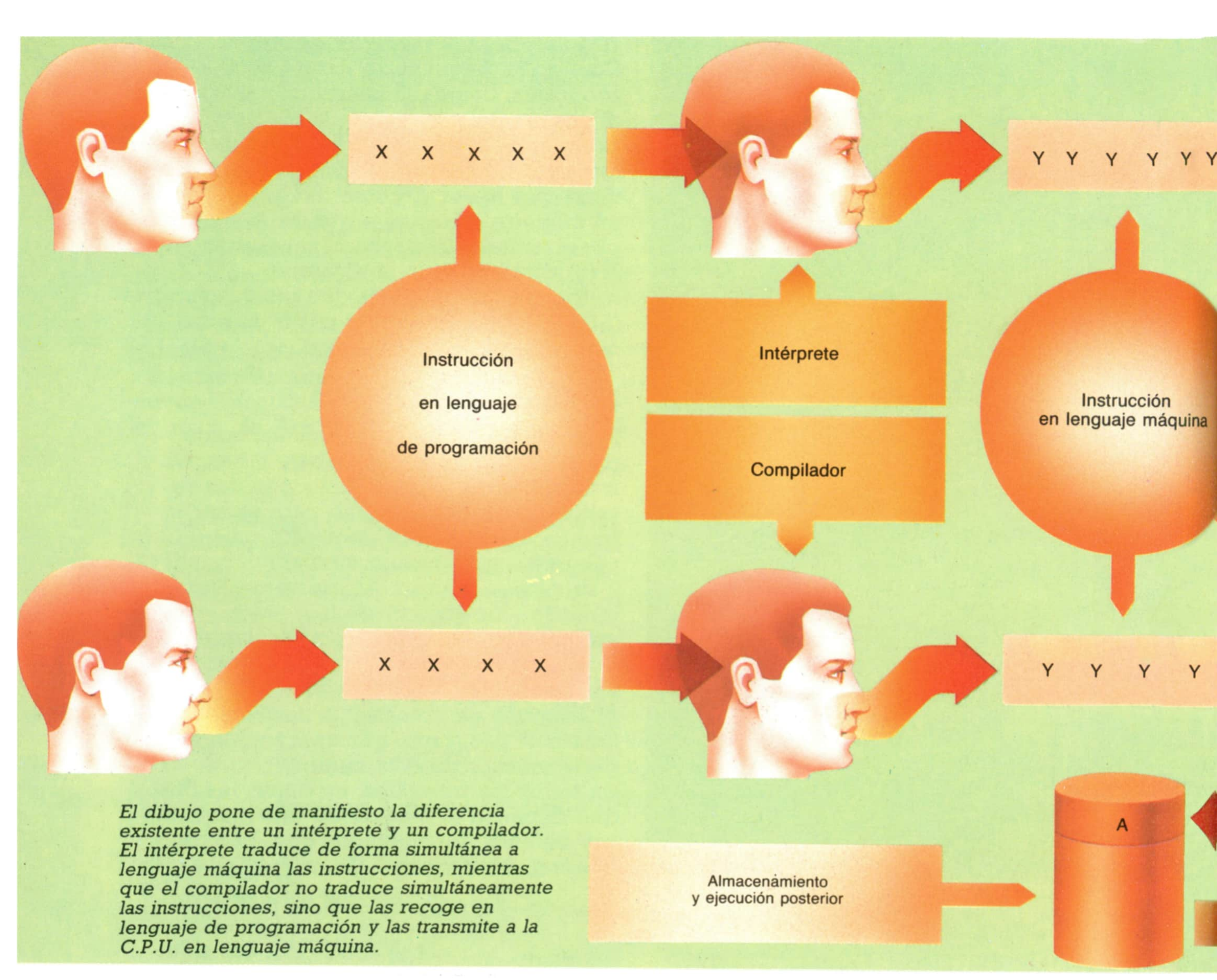



*¿Cómo opera la máquina de Arturo?* Cuando la máquina de Arturo recibe este programa, coloca la instrucción 0000 en la dirección de memoria 0, la 0001 en la 1 y así hasta la 6. Al recibir la orden de ejecutar, irá a la dirección 0, leerá lo que tiene que hacer y, como ha de preguntar un número, esperará a que se lo entremos por el teclado. Lo guardará en el acumulador A (0) y seguirá con la siguiente instrucción. Y así hasta que los cuatro unos (1111) le indiquen que a partir de este momento tiene que pararse.

## Lenguajes de bajo nivel

Programar con ceros y unos es, como puedes imaginarte, un trabajo laborioso y lento. Así que, cuando las máquinas ya sabían lo suficiente como para sumar lo que se les ordenaba, se buscó un programa que tradujera palabras sencillas a su código binario. Así nacieron los lenguajes de bajo nivel, los ensambladores, que no son más que lenguajes muy sencillos de traducción inmediata al código binario. Veamos ahora el lenguaje de la máquina de Arturo (véase el programa que aparece en la parte inferior de esta página).
En todos los manuales, es decir, los libros que explican el manejo de una máquina, hay muchas instrucciones que, por regla general, desconoces. Un manual es como

| *Código binario* | *Código escrito* | *Castellano* |
|---|---|---|
| ØØØØ | LDA | Carga |
| ØØØ1 | STA | Guarda |
| ØØ1Ø | ADD | Suma |
| ØØ11 | SUB | Resta |
| Ø1ØØ | AND | Pon uno si los dos son uno |
| Ø1Ø1 | ORA | Pon uno si uno es uno |
| Ø11Ø | JMP | Ve a |
| Ø111 | BZE | Ve si es cero |
| 1ØØØ | BNZ | Ve si es distinto de cero |
| 1ØØ1 | BMI | Ve si es menor que cero |
| 1Ø1Ø | BPL | Ve si es mayor que cero |
| 1Ø11 | LRS | Desplaza el acumulador a la izquierda |
| 11ØØ | NEG | Cambia el signo |
| 11Ø1 | INA | Recibe |
| 111Ø | OUT | Saca |
| 1111 | END | Para |



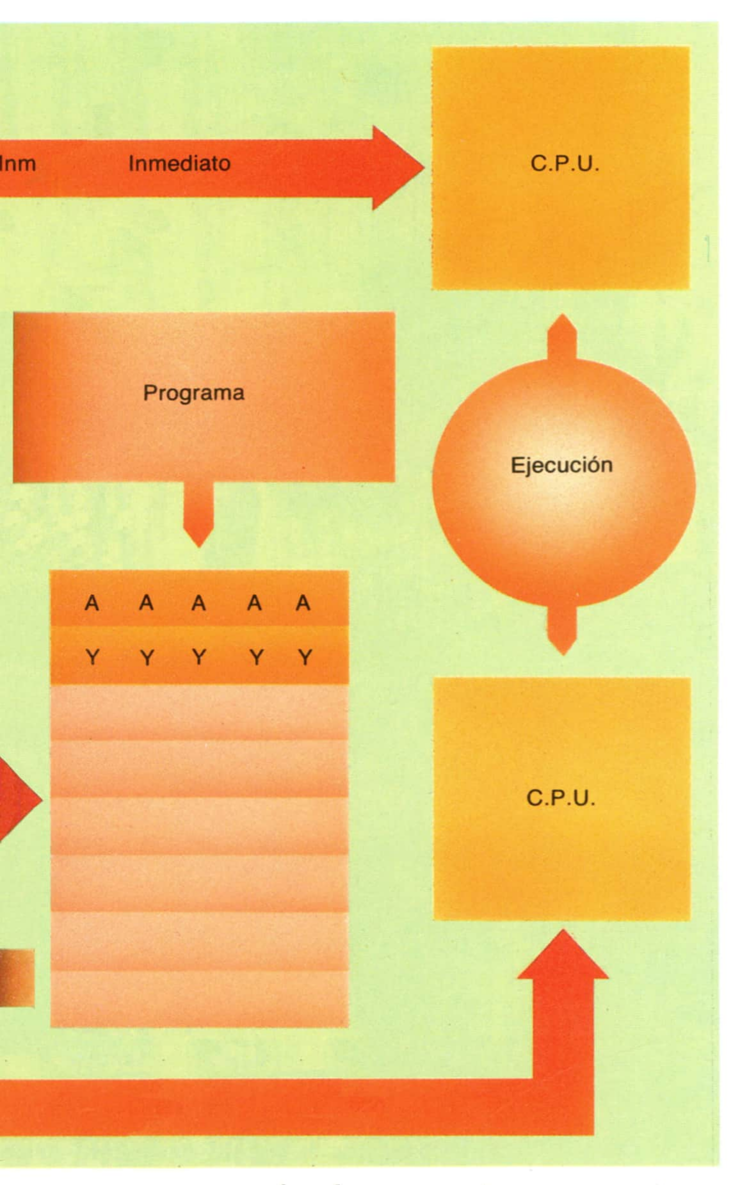

*El dibujo pone de manifiesto la diferencia existente entre un intérprete y un compilador. El intérprete traduce de forma simultánea a lenguaje máquina las instrucciones, mientras que el compilador no traduce simultáneamente las instrucciones, sino que las recoge en lenguaje de programación y las transmite a la C.P.U. en lenguaje máquina.*

un diccionario del idioma de la máquina y como tal lo debes utilizar: buscando lo que te interesa. Luego puedes leer las definiciones que no sabes, por si te pudieran ser útiles. Pero, como en todo diccionario, hay más palabras de las que conoces.
Con este lenguaje, nuestro programa queda de momento así:

```
Ø INA A
1 INA B
2 STA A 1ØØ
3 ADD B 1ØØ
4 OUT B
5 END
```

*¿Cómo lo entiende la máquina de Arturo?*
El programa de traducción de este lenguaje es muy sencillo. Sólo tiene que escoger la primera instrucción, buscar la palabra de 3 letras, ver cómo se traduce en 0 y 1. Protestar (dar un error), si no lo encuentra, traducir la A por un 0 y la B por un 1 y traducir el número a binario. Haz luego lo mismo con las siguientes instrucciones.
Éste es el primer paso en la traducción del lenguaje de los humanos al de las máquinas.

*Mejoremos el programa*
Tal como está hecho, este programa sólo tiene el inconveniente de que después de la primera suma, la máquina se para. Y




para que ésta haga otra, tienes que volver a enchufarla. Si queremos que cuando acabe vuelva a empezar, tenemos que utilizar una nueva instrucción: el Salta JMP de la máquina de Arturo. Debemos indicarle que vaya a una determinada dirección de memoria. Para ello, sólo tenemos que decírselo antes del END. Así pues, el programa quedará de este modo:

```
Ø INA A
1 INA B
2 STA A 1ØØ
3 ADD B 1ØØ
4 OUT B
5 JMP Ø          (la instrucción de ir a la
                          dirección Ø)
6 END
```

*Todavía una mejora más*
La ejecución de este programa plantea todavía el fallo de que no hay otra forma de pararlo que desenchufando la máquina, ya que nunca llegará a la instrucción END.
La solución radica en una instrucción que elija si debe seguir operando o parar la ejecución. Es la instrucción «SI» que ya conocemos. En la máquina de Arturo, tenemos cuatro maneras de decírselo. Que vaya a tal dirección si tiene un cero, o si no lo tiene, o si es mayor que cero o si es menor.
Elegiremos que se pare si el primer sumando es un cero. Después del INA A, intercalamos 3 BZE 7 y aumentamos en uno los números de las siguientes instrucciones; 3 BZE 7 quiere decir que, si lo que tiene en el acumulador A es un cero, debe ir a leer la dirección de memoria siete, donde encontrará el END y se parará.

## Los lenguajes de segundo nivel

El gran inconveniente de este lenguaje era que había que acordarse del lugar donde se guardan las cosas. Porque si para efectuar una suma hay que hacer tantas cosas, imagínate las que hubieran sido necesarias para enviar un cohete a la Luna. Así que hubo que buscar una solución. Se llegó a la conclusión de que podíamos decir primero cuántas direcciones de memoria nos iban a hacer falta, las bautizábamos como queríamos y le decíamos al robot que se las compusiera con ellas. Que se construyera una tabla que indicara que en la dirección tal, está la que se llama cual. Pero, para ello, antes de ejecutar el programa, había que traducir nuestro idioma al suyo. Se había inventado el compilador, es decir, un programa que sirve de intérprete entre nosotros y la máquina. Se habían inventado también las variables, que son el modo más sencillo de llamar a las direcciones de memoria: por su nombre en vez de por su dirección. Es como cuando, en lugar de decir la persona que vive en la casa 23.456, decimos Jorge, lo que, desde luego, es mucho más fácil. Siempre se han representado las variables como cajas, pizarras u hojas de papel donde puedes escribir y leer un único dato.

*Etiquetas*
También le diremos que numere las instrucciones y que si queremos referirnos a una en concreto, le pondremos un nombre, una «etiqueta», como se dice en «informático».
Así el programa para que Jane sepa sumar será el siguiente:

*Principio:*
PREGUNTA un número y guárdalo en la caja A
SI A es cero ve a fin
PREGUNTA un número y PONLO en B
SUMA A con B y PONLO en A
ENSEÑA A
Ve al principio
*Fin:*
PARA

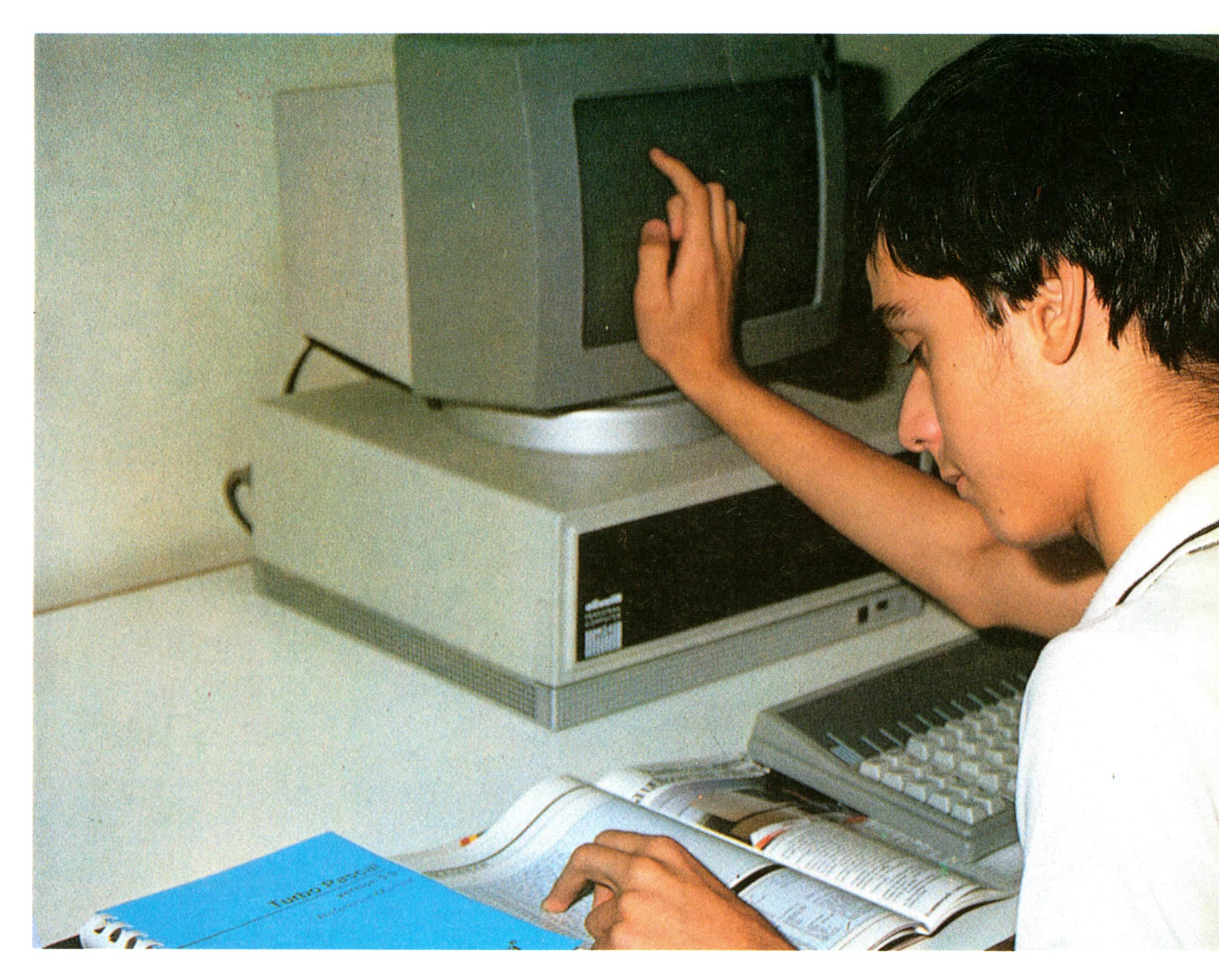
*Las ferias o exposiciones son una excelente ocasión para poner a prueba los conocimientos adquiridos en el campo de la informática, a la vez que permiten familiarizarse con las máquinas de tecnología más avanzada.*

Principio y fin son dos etiquetas y el compilador es el mismo que se encarga de decir en qué línea de instrucción (en qué dirección de memoria) están.
Para programar un ordenador, le diríamos en castellano:

*Principio:*
PREGUNTA A
Si A = Ø Ve a fin
PREGUNTA B
A = A + B
ENSEÑA A
Ve a principio
*Fin:*
PARA

*Asignaciones*
En este idioma es mucho más fácil hablarle a una computadora. La novedad que hemos introducido es A = A + B, en vez de SUMA A con B y PONLO en A. Esta forma de expresarlo, A = A + B, se denomina asignación y el signo = no significa una igualdad, sino la orden de que efectúe las






*El Turbo Pascal es un lenguaje de segundo nivel que ha sido diseñado para que el alumno vaya descubriendo y avanzando por sí mismo en campos tan dispares como la biología, las matemáticas, la lingüística o las ciencias sociales.*

operaciones de su derecha y las deposite en la variable de su izquierda.

*Compiladores*
Claro que antes de ejecutar este programa tendríamos que traducirlo al idioma de la máquina. Para ello, utilizaremos el programa compilador.
A partir del primer compilador, se pudieron ir desarrollando diversos lenguajes, es decir, nuevos traductores: cada lenguaje nuevo no es más que otro compilador. De este modo, un lenguaje puede hacer cada vez más cosas. Es como si tuviera más palabras.

## LENGUAJES DE PROGRAMACION

Existen muchos lenguajes de programación, aunque no tantos como los idiomas que utilizan los diversos pueblos del mundo. Como los lenguajes comunes, todos dicen lo mismo, pero de modo diferente. Observa cómo se dice «Pregúntame A» en distintos idiomas informáticos.

— *Fortran:*
READ (*,200) A. El Fortran es el primero de los idiomas de segundo nivel. READ significa leer en inglés. La * indica que el dato lo debe leer del teclado y el 200 que lo recibirá del modo que encontrará indicado en la etiqueta 200.

BS 2000

SISTEMA OPERATIVO

LENGUAJES

PASCAL
BASIC
A P L
FORTRAN
ALGOL
PL/1
RPG
COBOL

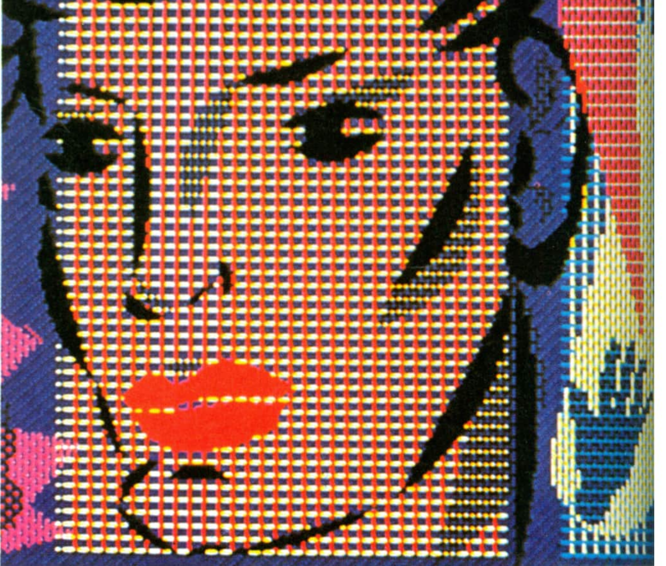
HERRAMIENTAS BASICAS DE SOFTWARE

UDS
SESAM
GOLEM
UTM
UDSC1
MRS
CODE 1
TOM
COLUMBUS

PROGRAMAS

INDUSTRIA
COMERCIO
SERVICIOS
BANCA
SEGUROS
Administración Central y Local
UTILIDADES
Diseño ayudado por ordenador
TRAFICO
NEGOCIOS
Educación y Ciencia
POLICIA

*Para realizar el precioso gráfico superior, que es la representación en pantalla de un programa de cartografía del espacio, o el dibujo con textura similar a los tejidos de lana de abajo, hay que aplicar un programa de diseño asistido por ordenador, realizado en alguno de los lenguajes de programación que se enumeran en el diagrama de la izquierda.*



— *Cobol:*
ACCEPT A. Parece más fácil que el Fortran, pero antes debes haber descrito la variable A con pelos y señales. Mucho más que lo que tienes que explicar en la etiqueta 200, si programas en Fortran.

— *Pascal:*
READ (A). También tienes que haber definido antes la variable A.

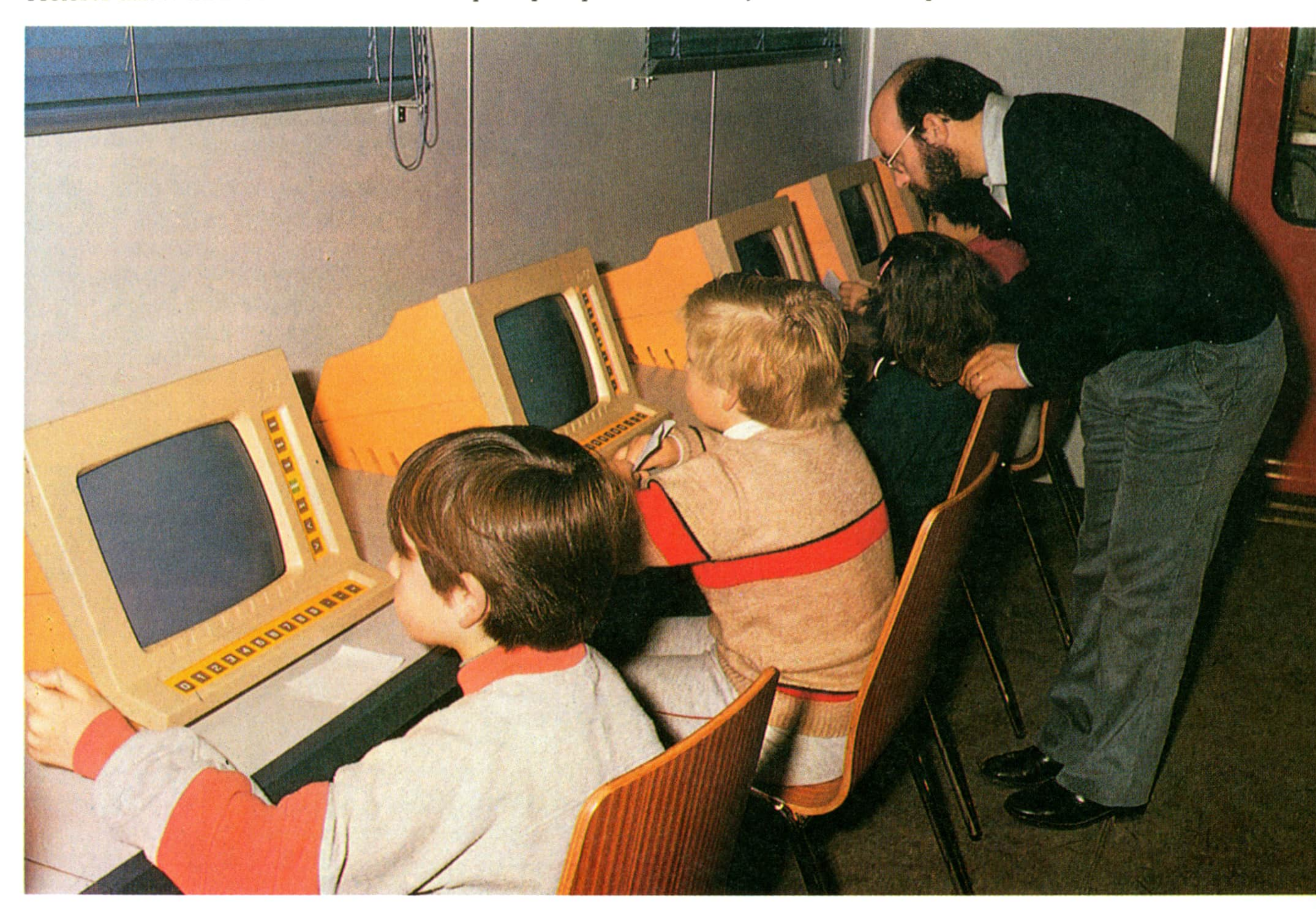
— *Basic:*
10 INPUT A. Aquí no hay que definir la A, pero se deben numerar todas las instrucciones.

Para que el compilador sepa cuánta memoria debe reservar para cada una de las variables, cuántas hojas del bloc o el tamaño de las cajas, según el ejemplo que prefieras, debemos fijar unas reglas.

## Bytes

La primera regla es fijar una unidad mínima de información. Puesto que en un bit caben sólo dos posibilidades, se pensó en tomar los bits en grupos, como hicimos antes con las palabras.
Se decidió que con ocho bits se tenían 256 combinaciones de ceros y unos. La reunión de 8 bits recibe el nombre de byte. Pronúncialo *bait,* puesto que es inglés.
La información que debemos guardar en memoria estará compuesta de letras y números. Necesitaremos las 27 letras del alfabeto, en mayúsculas y minúsculas, los diez números y los signos de puntuación. También la tecla para borrar, la de la barra de espacios y otras especiales de los ordenadores. Con 7 bytes sólo había 128 posibilidades, lo que es poco, y con 9, 512, que son demasiadas.

*Profesor dando instrucciones a los alumnos para que aprendan a manejar un ordenador personal.*





Así que se eligió 8 bits, que además es un número muy divisible y potencia de 2, que es a lo que tenemos que ir a parar.

*Las famosas «K»*
Ya contamos pues con un byte. La capacidad de memoria de una computadora se mide en bytes. Pero puesto que un byte es muy poco, se aceptó el Kbyte, expresado con una K, es decir 1.024 bytes, como unidad de memoria. Para entendernos, diremos que en una K caben 1.024 letras. No creas que es mucho. Si escribes una carta a máquina, te cabrán 2.100 letras en cada página.

## Tipos de variables

Una vez que ya tenemos la unidad de medida, vamos a explicar lo que tiene que hacer un ordenador. Deberá acordarse de letras y números. Y ello porque con los números se debe poder operar y con las letras no. Si le preguntamos a Jane cuánto es una mesa por una bicicleta, se quedará algo sorprendida. Y tendrá razón. Para evitarlo, le diremos que hay dos clases de variables: las que permiten operar, que llamaremos numéricas, y las que no lo permiten, que denominaremos alfabéticas. Ten en cuenta que aunque dentro de una variable alfabética haya un número, no se podrá operar con él.

*Variables numéricas*
Los alfabetos no plantean problemas. Tantas letras tiene el dato de largo, tantos bytes ocupará. Pero con los números la cosa se complica. ¿Cómo será de grande el número que voy a guardar? ¿Puede ser más grande que mil, que un millón, que mil millones? La primera reacción ante el problema fue curarse en salud y decir que todas las variables numéricas serían grandísimas. Se cogen ocho bytes y se tienen $2^{64}$ posibilidades de bombillas encendidas y apagadas. O se coge un byte por dígito previsto. Estamos hablando de una época en la que había máquinas grandes y muy caras, casi en la prehistoria de los ordenadores.

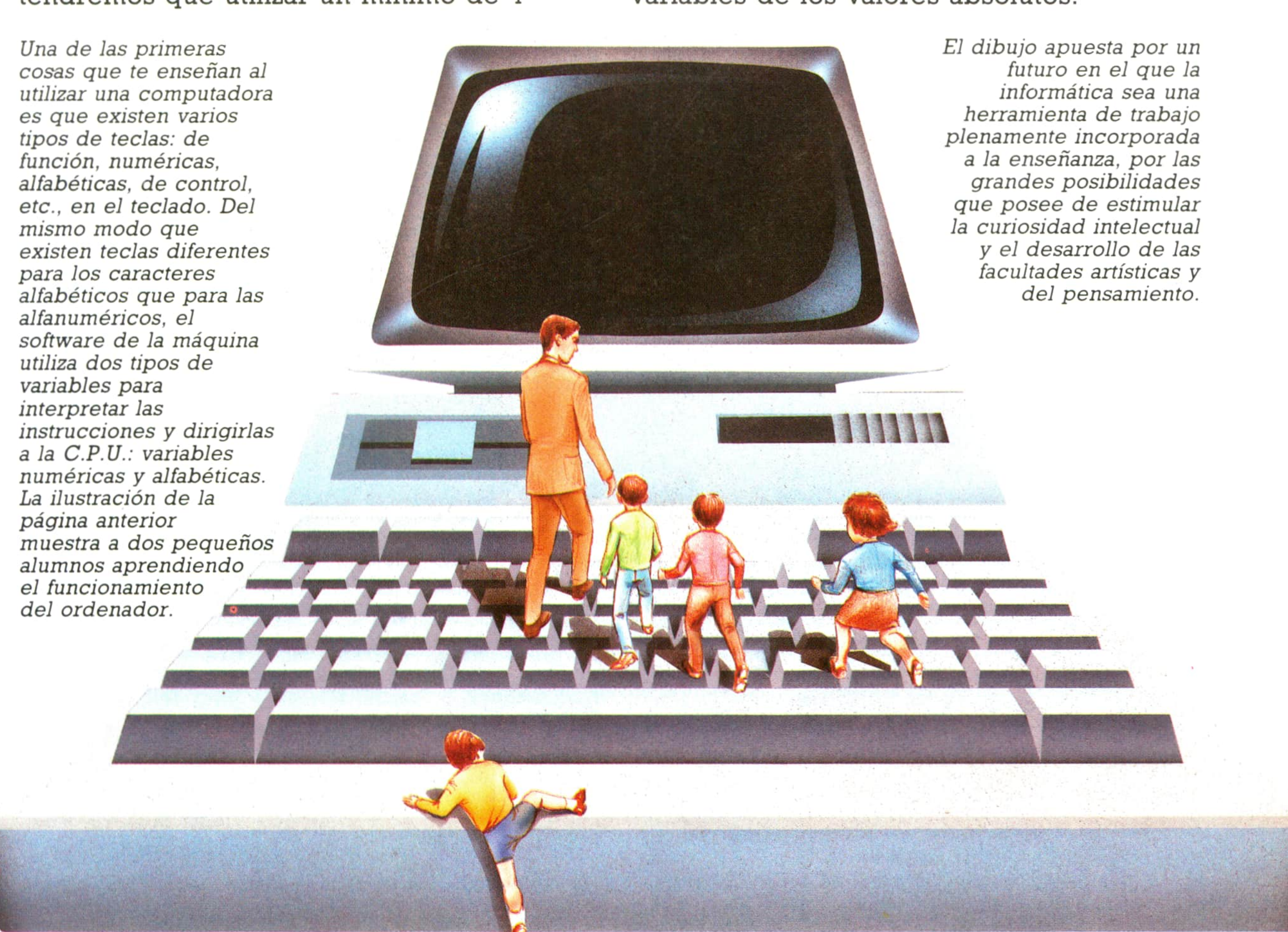
Pero uno y otro sistema eran un despilfarro de memoria. Y la memoria, al principio, era muy cara; no olvides que es lo más importante de todo. Así que se decidió dividir las variables numéricas según su tamaño.

*Enteros*
Si cogemos dos bytes, tenemos $2^{16}$ combinaciones posibles, lo que representa 65.235 valores posibles. Si nos comprometemos a no usar este tipo de variable más que entre valores enteros desde —32.627 hasta +32.627, con dos bytes podemos guardar números de hasta 5 dígitos (siempre que no pasen de 32.627). A este tipo de variable lo llamaremos *entero.* No sirve para guardar decimales, ni para valores muy grandes, pero es muy útil. Y representa un buen ahorro de memoria.

*Reales*
Si necesitamos trabajar con decimales, tendremos que utilizar un mínimo de 4 bytes para asegurarnos seis dígitos con una coma en cualquier sitio. A las variables que aceptan decimales las llamamos *reales.*
Esta división entre variables depende, evidentemente, del lenguaje. El Basic tiene tres divisiones (2, 4 y 8 bytes), el Pascal dos (4 y 8), el Turbopascal dos (2 y 6) y el Fortran, enteros de 2 y 4 y reales de 4 y 8.

## Indicadores

Para representar las variables, se emplean letras o palabras, que llamamos indicadores. Nos sirven de traducción entre nosotros y el lenguaje máquina. Para asignar nombres a las variables, hay que seguir unas reglas. En primer lugar, no pueden ser instrucciones. Si le dices "Pregunta", el traductor debe saber traducirlo por la rutina correspondiente, no reservando memoria. Los indicadores deben empezar por una letra. Así para el compilador es más fácil distinguir las variables de los valores absolutos.



*Una de las primeras cosas que te enseñan al utilizar una computadora es que existen varios tipos de teclas: de función, numéricas, alfabéticas, de control, etc., en el teclado. Del mismo modo que existen teclas diferentes para los caracteres alfabéticos que para las alfanuméricos, el software de la máquina utiliza dos tipos de variables para interpretar las instrucciones y dirigirlas a la C.P.U.: variables numéricas y alfabéticas. La ilustración de la página anterior muestra a dos pequeños alumnos aprendiendo el funcionamiento del ordenador.*

*El dibujo apuesta por un futuro en el que la informática sea una herramienta de trabajo plenamente incorporada a la enseñanza, por las grandes posibilidades que posee de estimular la curiosidad intelectual y el desarrollo de las facultades artísticas y del pensamiento.*



*El BASIC es un lenguaje de alto nivel, muy difundido en la actualidad, por su fácil manejo y aprendizaje. Es una buena herramienta para realizar trabajos escolares o participar en juegos.*

# BASIC

El lenguaje Basic (siglas de *Beginer's All-purpose Symbolic Instruction Code,* es decir, Código simbólico de instrucciones para principiantes con todos los propósitos) constituyó la invención de un nuevo sistema de traducción.
Hasta entonces, el mecanismo de la programación consistía en escribir un programa en un papel, perforar las fichas correspondientes, meterlo en la máquina, compilarlo y ejecutarlo.
Es decir, hacían falta, como mínimo, dos programas, uno que lo metiera en la máquina y otro que lo tradujera. El Basic es un único programa, que cumple ambas funciones. Por un lado, le das las instrucciones y, por otro, las traduce y las ejecuta. Pero en vez de traducirlas todas juntas, lo hace una a una. A este tipo de compilador le llamaremos *intérprete,* porque traduce nuestro idioma al de la máquina. Posee la ventaja de que estás hablando constantemente con la máquina y el inconveniente de que siempre tiene que traducir, con lo que es mucho más lento.

## Variables Basic

En principio, todos los Basic diferencian los dos tipos de variables, numéricas y alfabéticas. Para indicar que una variable es alfabética se termina siempre por el signo $. Así A$, B$ indican que las variables son alfabéticas. A, D son variables numéricas.
En Basic no hay que dimensionar las variables. El intérprete se ocupa de hacer el direccionamiento de memoria a medida que van saliendo. Para ello, antes de actuar, comprueba si ya han aparecido antes en el programa. En caso afirmativo, las remite a la dirección de memoria que ya ha calculado, mientras que si son nuevas, les adjudica una nueva.
Los números se distinguen de modo diferente según la máquina. Ya se ha explicado antes. Si utilizas el sistema operativo MS-DOS, las variables enteras acaban en % (por ejemplo, A%), los reales de 4 bytes en ! o en nada (B! o B) y los de doble precisión u 8 bytes en # (D#).

## Palabras del Basic

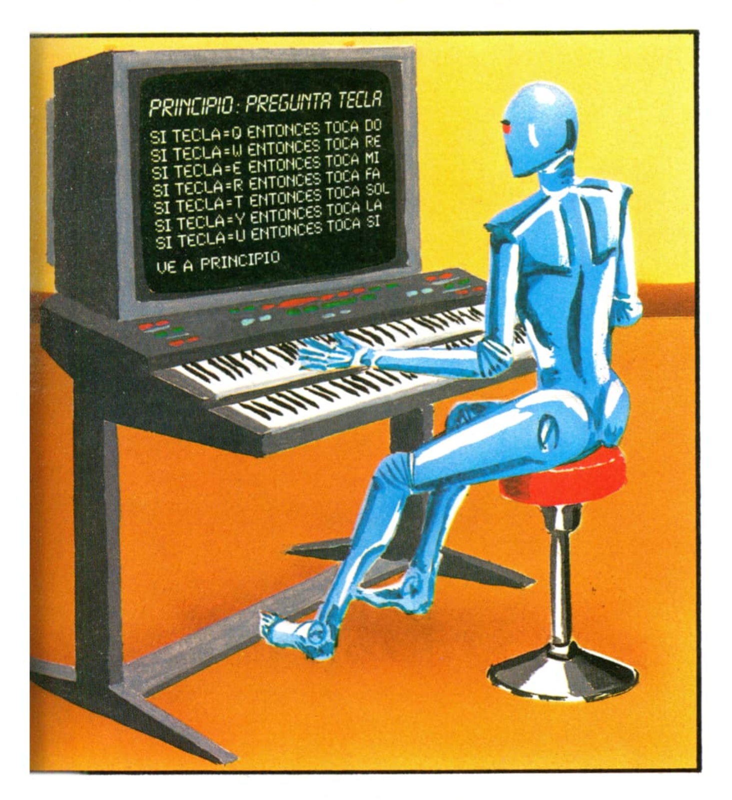
Cuando aprendas cualquier lenguaje, tienes que pensar que se trata sólo de traducir a la máquina en cuestión lo que le dirías a Jane. Esa es la utilidad del manual. Dado que los ejemplos los vamos a escribir en Basic, ya te indicaremos cómo se escribe a medida que vayan saliendo. Cada máquina tiene su propio programa de Basic. Se parecen en lo fundamental, pero se diferencian en los detalles. Reciben incluso nombres distintos.

## Algunos ejemplos

Vamos a hacer varios programas. Pero antes queremos advertirte que intentar aprender a programar sin máquina es como pretender aprender a nadar sin agua. Dando por supuesto que dispones de un ordenador, empezaremos enchufando la máquina. Como sabes, al hacerlo se carga la memoria ROM, que es la que no se borra nunca. Según la máquina que tengas, el Basic puede estar ya en la ROM; así que no deberás hacer nada. Si no es así, tendrás que llamarlo. Si tu máquina utiliza el sistema operativo MS-DOS, debes poner el diskette del sistema operativo, que te entregaron con la máquina, en la disquetera A. A continuación, el ordenador te preguntará el día y la hora. Una vez hecho esto, la máquina está lista para trabajar y debes

ejecutar el programa Basic, al que se puede llamar de varias formas, según la marca de tu computadora. Para llamarlo, simplemente lo escribes y pulsas RETURN (o INTRO o ENTER), que es la tecla que significa el «¡ar!» de los militares o el «¡ya!» de los profesores de gimnasia.

*Cargar el Basic*
Ya estás en Basic, es decir, estás en un programa que entiende muchas cosas que tú le puedes decir. Son las instrucciones. Puedes decirle, por ejemplo, "PRINT Pepe" y al pulsar INTRO verás que la palabra "Pepe" aparece justo debajo de tu orden.
Para que la orden permanezca en la memoria, debes poner delante el número de instrucción: "10 PRINT Pepe".
Ahora no aparece la palabra "Pepe". Pero si escribes LIST y lo ejecutas, te saldrá esta instrucción. LIST es una orden que te lista el programa que tienes en memoria. Para borrar el programa, basta con ordenar NEW (nuevo). Para grabar un programa en el diskette (o en el cassette), le debes decir SAVE "/nombre del programa/" (así, entre comillas) y para cargarlo en memoria, lo haces con LOAD "/nombre del programa/". Para ejecutarlo, debes ordenarle RUN. SAVE es salvar, LOAD es cargar y RUN es correr, en inglés, naturalmente.



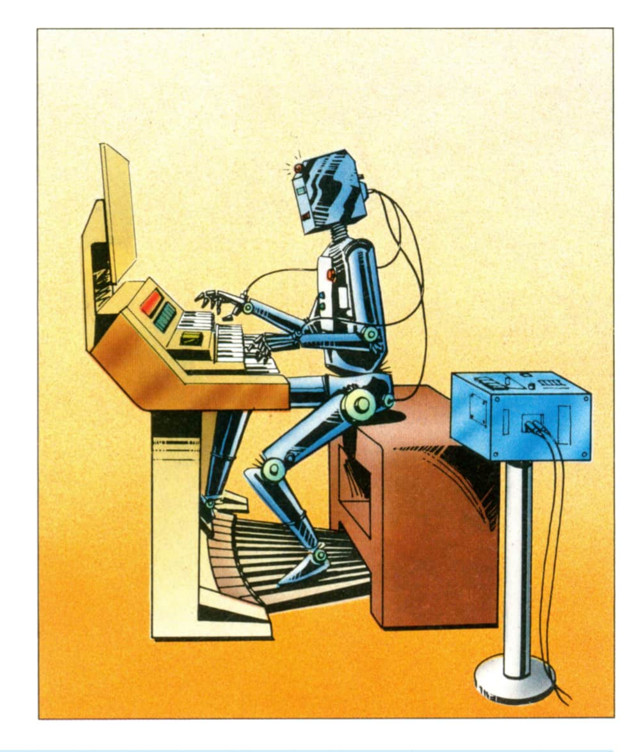

*Jane está empezando a desarrollar un programa de música para tocar el piano. Con el texto que aparece en pantalla, Jane habrá conseguido establecer las equivalencias entre las teclas del ordenador y las notas musicales del pentagrama.*

*Programar en Basic*
Para empezar a programar en Basic, sólo te falta saber que, si bien cada instrucción debe ir numerada, puedes poner varias en la misma línea, siempre que las separes con dos puntos (:). Otra instrucción que debes conocer es el REM, que significa que no hagas nada y que sirve para ir explicando lo que va haciendo el programa. Por último, puedes programar en mayúsculas o minúsculas, pues al intérprete de Basic le da lo mismo: él las pone en mayúsculas.

*Programa para sumar*

```
1Ø INPUT A: REM
(es la instrucción METE en la variable
A)
2Ø IF A=Ø THEN GOTO 7Ø: REM
(si A=Ø ve a la instrucción 7Ø)
3Ø INPUT B: REM
(METE en la variable B)
4Ø a=a+b: REM
(suma a con b y mételo en la variable a)
5Ø PRINT a: REM
(escribe el valor de la variable A)
6Ø GOTO 1Ø: REM
(ve a la instrucción 1Ø)
7Ø END: REM (que se pare)
```

*Imitemos un piano*
Siempre que tengas que hacer un programa, tienes que empezar por pensar cómo le puedes pedir a Jane que te lo haga. Un piano no es más que varias teclas, cada una de las cuales tiene su sonido. Teclas no te faltan en el teclado; así que sólo tienes que decidir qué letra corresponderá a un do o a un mi. Si coges la fila de las letras QWERTYU y decides que Q=do, W=re, E=mi, R=fa, T=sol, Y=la, U=si, sólo tienes que decir, lo que muestra la pantalla del dibujo de la izquierda.
Cuando pulses una de estas teclas, Jane obedecerá y luego volverá a preguntar



*Prototipo de robot pianista presentado en la Feria Internacional de Tsukuba (Japón), en 1985.*

qué otra tecla. Si no tocas ninguna o pulsas una que no corresponde a éstas, la máquina volverá a preguntarte qué otra tecla.

*La instrucción "PLAY"*

Vamos a pasarlo a BASIC. Coges el manual y miras cómo se escribe la orden de tocar. "PLAY A" y toca un LA, porque en inglés la escala musical se escribe C-D-E-F-G-A-B. El manual también señala que los sostenidos se indican como "PLAY A#" (LA sostenido) y los bemoles como "PLAY A-" (LA bemol). También explica cómo cambiar de octava "PLAY >" o "PLAY <", lo que señala la escala superior o inferior. Esto se puede hacer también indicando la "Octava PLAY Ø2 A", un LA de la segunda octava.

```
 1Ø REM SAVE "piano. bas": REM (es conveniente empezar siempre con el nombre del
    programa)
 2Ø k$=INKEY$: REM (inkey$ es como un input, pero sin tener que pulsar INTRO, y
    sólo coge un carácter)
 3Ø IF k$="" THEN 2Ø: rem (si no has pulsado nada, que vuelva a esperar)
 4Ø IF k$="Q"  THEN PLAY "C": GOTO 11Ø: REM toca DO
 5Ø IF k$="W" THEN PLAY "D": GOTO 11Ø: REM toca RE
 6Ø IF k$="E"  THEN PLAY "E": GOTO 11Ø: REM toca MI
 7Ø IF k$="R"  THEN PLAY "F": GOTO 11Ø: REM toca FA
 8Ø IF k$="T"  THEN PLAY "G": GOTO 11Ø: REM toca SOL
 9Ø IF k$="Y"  THEN PLAY "A": GOTO 11Ø: REM toca LA
1ØØ IF k$="U"  THEN PLAY "B": GOTO 11Ø: REM toca SI
11Ø k$="": REM (es conveniente limpiar las variables después de utilizarlas)
12Ø GOTO 2Ø: REM (ve a la 2Ø)
```

*Ampliemos la escala*

Si quieres ahora indicar los bemoles y los sostenidos para señalar las teclas, puedes, como en el piano, ponerlas en su lugar de la escala, pero por arriba. El 2 será el DO#, el 3=RE#, el 4 no será nada (el MI sostenido es el FA), el 5=FA#, el 6=SOL#, el 7 el LA# y también el SI#=DO. Ahora sólo te falta escribir las nuevas instrucciones. Lo más lógico es intercalarlas:

*Aumentemos las escalas*

Si quieres tener dos escalas, define las teclas ZXCVBN como la siguiente escala y SDGHJ como sus sostenidos. Tienes, para ello, que decir en qué escala debe tocar:

```
45 IF k$="2" THEN PLAY "C#":
   GOTO 11Ø: REM (toca DO sostenido)

Y así la 55, 75, 85, 95, 1Ø5.
```

```
4Ø IF k$="Q" THEN PLAY "Ø2 C":
   GOTO 11Ø: REM (toca DO en la escala 2)
42 IF k$="Z" THEN PLAY "Ø3 C":
   GOTO 11Ø: REM (toca DO en la escala 3)
45 IF k$="2"  THEN PLAY "Ø2 C#":
   GOTO 11Ø
47 IF k$="S"  THEN PLAY "Ø3 C#":
   GOTO 11Ø
```

Para tener más escalas, puedes también asignar una tecla para subir una escala y otra para bajarla. P=subir, Ñ=bajar, por ejemplo.

Para ello debes decir entonces que:

```
32 IF k$ = "P" THEN PLAY ">"
33 IF k$ = "Ñ" THEN PLAY "<"
```

Claro que esto no sirve de mucho, pues tú le indicas que toque en la octava 2 o en la 3. Para ello deberíamos darle un valor variable a la octava. Entonces quedaría así:

```
15 octava = 2
20 IF k$ = INKEY$
30 IF k$ = "" THEN 20
32 IF k$ = "P" THEN
   octava = octava + 1:IF octava > 6 THEN
   octava = 6: REM (no se puede pasar de
   la octava 6)
33 IF kp$ = "Ñ" THEN octava = octava
   —1:if octava < 0 then octava = 0
40 IF k$ = "Q" THEN PLAY "0" + MID$
   (STR$(octava),2,1) + "C": goto 110
42 IF k$ = "Z" THEN PLAY "0" + MID$
   (STR$(octava + 1),2,1) + "C": goto 110
```

## Otras instrucciones nuevas

MID$ (/variable alfabética/, /byte de origen/, /número de bytes/

Esto quiere decir que si tienes

a$ = "abcdefghijk"
b$ = MID$ (a$,3,4) (debes poner dentro de la variable b$ 4 bytes de a$ a partir del byte 3; es decir, si haces PRINT b$ te saldrá: "cdef")
STR$ (/variable numérica/) (convierte la variable numérica en alfabética; como le pone un blanco delante, hay que coger sólo a partir del segundo byte)

También tienes de nuevo el signo "+" entre dos variables alfabéticas, cuya misión es poner la segunda detrás de la primera. Si suponemos que octava = 2, la instrucción 40 quedará así:

```
PLAY "0" + "2" + "C"
(es decir, PLAY "02 C", o sea,
decir TOCA DO en la escala 2)
```

*Una solución más cómoda*
Este programa tiene el inconveniente de que para tocar el "si" de la segunda escala, ha tenido que sondear muchas condiciones, lo que se nota, aunque el ordenador sea muy rápido, y nuestro piano no es precisamente perfecto. Sin embargo, casi todo tiene remedio. Así es como se hace:

```
10 REM SAVE "piano.bas"
15 octava = 2
17 e$ =
"Q2W3ER5T6Y7UZSXDCVGBHNJMPÑ":
REM
   (le dices todas las teclas posibles que
   puede aceptar)
20 k$ = INKEY$
30 IF k$ = "" THEN 20
40 I = INSTR(E$,k$): REM
   (te dice qué posición ocupa k$ dentro
   de la variable E$; si no estuviera te
   daría i = 0)
45 IF i = 0 THEN 300: REM
   (no se ha pulsado ninguna tecla
   operativa)
46 IF k$ = "P" THEN
   octava = octava + 1:IF octava > 6
   THEN octava 6
47 IF k$ = "Ñ" THEN
   octava = octava—1:IF
   octava < 0 THEN octava = 0
50 ON I GOTO 60, 70, 80, 90, 100, 110,
120, 130, 140, 150, 160, 170, 180, 190,
200, 210, 220, 230, 240, 250, 260, 270,
280, 290, 300, 300:REM
   (le dices que si I = 1,
   vaya a la 60, si i = 2 a la 70
   y así sucesivamente; como I sólo
   puede valer de 0 a 26,
   lo tienes previsto)
60 PLAY "O" + MID$(STR$(octava),
2) + " C":
GOTO 300: REM (que toque DO en la
   escala indicada por el valor de la
   variable octava)
70 PLAY "O" + MID$ STR$(octava),
2) + "C$": GOTO 300
   Y así sucesivamente hasta llegar a la
   instrucción:
300 k$ = "":REM (limpiar la variable)
310 GOTO 20: REM (vuelta a empezar)
```

*Un progreso más corto*
Tal vez te dé pereza escribir todo este programa. Es normal y no eres un caso único. Debemos pensar entonces si no habría una forma más corta de hacerlo. La hay. Fíjate que todas las instrucciones se parecen mucho. Sólo se diferencian en la nota que deben ejecutar.

```
10 REM SAVE "piano.bas"
20 N$ = "C C#D D#E F F#G G#A A#B
B#": REM
(son las 12 notas que debe tocar)
30 octava = 2
40 e$ = "Q2W3ER5T6Y7UZS
XDCVGBHNJMPÑ": REM
(le dices todas las teclas posibles que
puede aceptar)
50 k$ = INKEY$
55 IF k$ = "" THEN 50
60 I = INSTR(E$,k$): REM
(ya sabes qué tecla se ha pulsado)
70 IF i = O THEN 140
80 IF k$ = "P" THEN
octava = octava + 1:if octava > 6 THEN
octava = 6
90 IF k$ = "Ñ" THEN octava = octava-
1:IF octava < 0 THEN octava = 0
100 IF i > 24 THEN 140: REM
(si sólo es un cambio de escala, no debe
hacer nada)
110 IF i < 13 THEN suma = 0 ELSE
suma = 1 :i = i—12:REM
(si la nota es de la segunda escala, hay
que restarle 12 porque sólo tenemos 12
sonidos posibles)
115 i = 2*i—1:REM
(como las notas ocupan dos bytes, debes
establecer una correspondencia entre las
variables N$ y E$; la 1 es la 1, la 2 es la
3, la 3 es la 5; siempre será el doble
menos 1; el signo "*" indica
multiplicación)
120 PLAY
"0" + MID$(STR$(octava + suma),
2) + MID$(n$,i,2)
130 k$ = ""
140 GOTO 50
```

## El ordenador puede hacerte los deberes

En principio, puede parecerte que es un programa muy difícil. Pero lo que ocurre, en realidad, es que necesitas muchos programas, tantos como tipos de deberes.

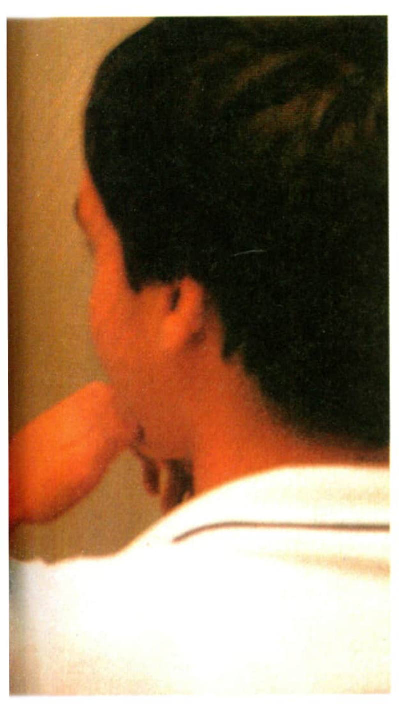


Así que lo primero que debes hacer es elegir por cuál quieres empezar. Supongamos que por la geometría, que es el más fácil. Y dentro de la geometría, empezaremos por las áreas:

```
10 REM SAVE "areatri.bas": REM
(siempre es la primera instrucción)
```

*Calcular el área de un triángulo*
Imaginemos que empezamos por las áreas del triángulo. Sabes que la superficie es S=b×h/2 (base por la altura dividido por dos). Deberás, pues, entrar la base y la altura, hacer la operación y enseñar el resultado. Es casi igual que cuando hicimos el programa de la suma.

```
20 PRINT "entra la base"
30 INPUT B:IF b = 0 THEN END: REM
(ya sabes, por la suma, lo que hay que
hacer)
40 PRINT "entra la altura"
50 INPUT h
60 s = b*h/2: REM
(el * es la multiplicación y la / es el
signo de la división)
70 PRINT "la superficie es" ;S
80 GOTO 20: REM
(vuelve a pedir otra)
```



*A la izquierda, joven operando con un programa Turbo Pascal para la realización de problemas de geometría. Abajo, menú en pantalla que muestra las diversas posibilidades que tienes de aprovecharte del ordenador para hacer los deberes.*

*Escoger un menú*

Del mismo modo, puedes hacer un programa para cada problema de geometría. Pero tener tantos programas es, en realidad, bastante inútil; así que puedes ponerlos todos juntos y sólo tienes que elegir el que quieras hacer. Se trata de realizar lo que se llama un "menú", que viene a ser como los menús de los restaurantes: decirle a la computadora qué es lo que quieres que te haga. Verás:

```
10 REM SAVE ''areas.bas''
20 PRINT ''este programa calcula areas
   de''
30 PRINT ''cuadrados... 1'': REM
40 PRINT ''triángulos... 2'': REM
50 PRINT ''círculos... 3'': REM
60 PRINT ''trapecios... 4'': REM
70 PRINT ''esferas... 5'': REM
```

Puedes poner todos los que quieras. Sólo tienes que tener en cuenta que en la pantalla sólo caben 24 líneas; así que si quieres calcular más de 24 cosas, debes partir la pantalla en dos, o en tres; también puedes calcular volúmenes y todo aquello que sea la simple aplicación de una fórmula.

Después de estas instrucciones, debes entrar la opción que desees.

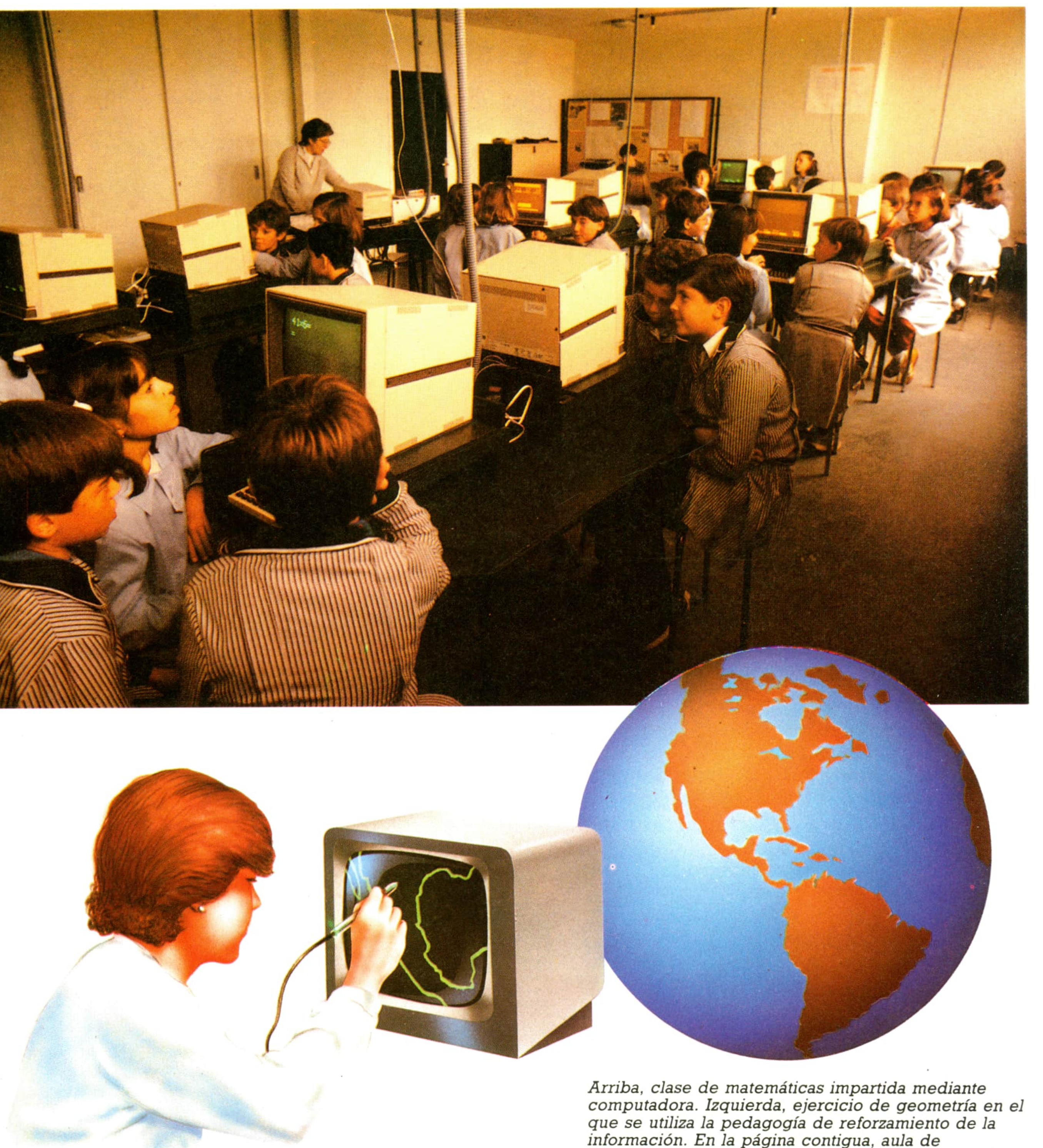
```
80 PRINT ''entra el numero de la
opcion''
90 INPUT opcion
100 IF opcion <0 OR opcion>5 THEN
20: REM
(esto es el sondeo, para que no le pidas
una opción que no has previsto)
110 IF OPCION = 0 THEN END: REM
(que se pare; has acabado)
120 ON opcion GOTO 200, 300, 400,
500, 600: REM (que se vaya donde
corresponda)
200 REM cuadrados
(para que tú sepas qué hace aquí)
210 PRINT ''entra el lado''
220 INPUT l
230 IF l = 0 THEN 20: REM (que se
vuelva al menú)
240 s=l ^2 REM (así se le pide que
coloque el cuadrado)
250 PRINT ''la superficie es'' ;S
260 GOTO 210
```

*Calcular el área del círculo*

Ya hemos hecho el triángulo; así que sólo tienes que escribirlo cambiando sus números de instrucción. El círculo sería de este modo:

```
400 REM circulos
410 PRINT ''entra el radio''
420 INPUT radio
430 IF radio = 0 THEN 20
440 s=3.141598*r ^2
450 PRINT ''la superficie es'' ;S
460 GOTO 410
```





*Arriba, clase de matemáticas impartida mediante computadora. Izquierda, ejercicio de geometría en el que se utiliza la pedagogía de reforzamiento de la información. En la página contigua, aula de informática para el aprendizaje de ciencias sociales.*

Te sugiero que hagas todos los que se te ocurran. Puedes tomártelo como si se tratara de unos deberes que, sin embargo, sólo los tendrás que hacer una vez, porque la computadora se acordará siempre.

## Geografía

Vamos a enseñarle a la máquina las capitales. Pero como no las sabe, hay que enunciárselas primero.

```
10 REM SAVE "capital.bas"
20 DATA España, Mexico, Japon,
Honduras, Republica Federal de
Alemania, Estados Unidos de America,
Argentina, Francia, China, Italia: REM
(DATA es una instrucción para darle
datos fijos a un programa; más
adelante, te explicaremos cómo se usa)
30 DATA Madrid, Mexico DF, Tokio,
Tegucigalpa, Bonn, Washington, Buenos
Aires, Paris, Pekin, Roma: REM (como
puedes ver, los países y sus capitales
están en el mismo orden)
35 numero = 0
40 PRINT "entra el pais"
50 INPUT pais$
60 IF pais$ = "" THEN END
70 RESTORE 20:REM (que se prepare a
leer los datos de la instrucción 30)
40 FOR i = 1 to 10: REM
(sólo sabe 10 capitales)
```

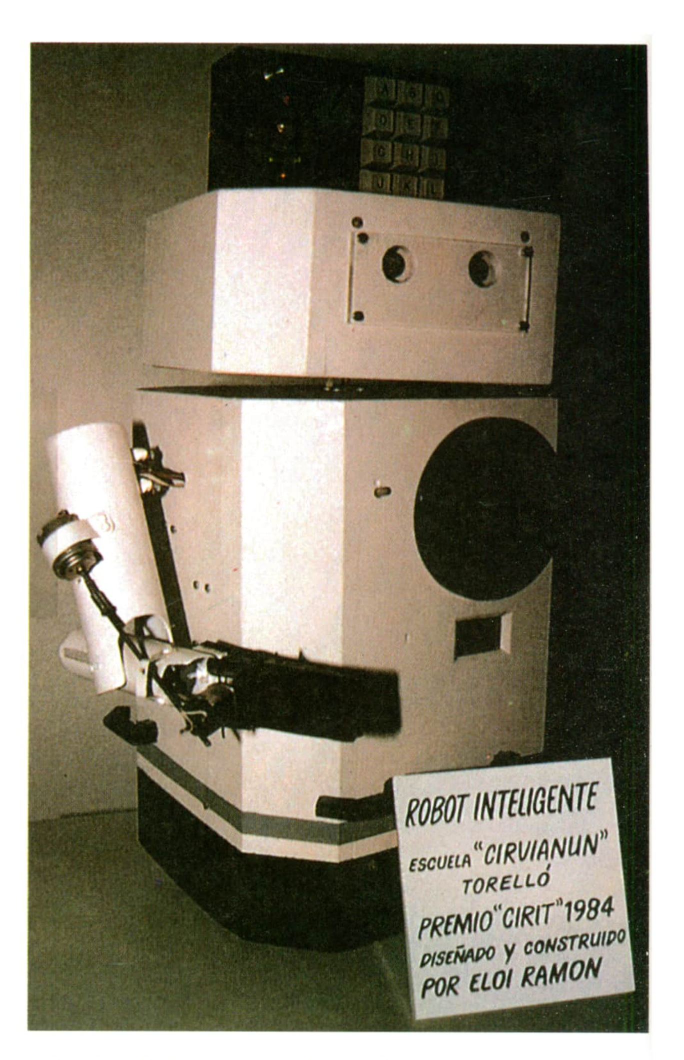
*La instrucción FOR NEXT*
Ésta es la instrucción repetitiva más usada en BASIC. Es el HAZ que tantas veces le enseñamos a Jane. La variable I irá aumentado de valor hasta llegar a su máximo permitido, 10 en este caso. La máquina ejecutará de esta manera todas las instrucciones hasta que encuentre una que ponga NEXT I.

```
50 READ p$:REM (que ponga en p$ el
valor de la data correspondiente)
60 IF pais = p$ THEN numero = i:REM
(que el país es el I)
70 NEXT i:REM (señal de final del bucle
o conjunto de instrucciones que debe
repetir)
80 IF numero = 0 THEN PRINT "no lo
se": GOTO 40:REM (que diga que no lo
sabe porque no se lo has dicho antes)
90 RESTORE 30:REM
(que se prepare a leer las capitales)
100 FOR i = 1 TO numero:REM
(que tiene que hacerlo hasta el valor
número)
110 READ capital$:REM
(que vaya leyendo capitales y las ponga
en capital$)
120 NEXT i
130 PRINT "la capital es"; capital$
140 GOTO 35:REM
(que te vuelva a preguntar un país)
```





*Un grupo de alumnos en una clase de matemáticas por computadora.*



*Robot diseñado y construido por los propios escolares.*

*Invertir el programa*
Si quisieras hacer el programa inverso, es decir, que dada una capital el ordenador te conteste a qué país corresponde, sólo tendrías que cambiar las instrucciones de los RESTORE y, naturalmente, los textos de la pregunta y la respuesta.

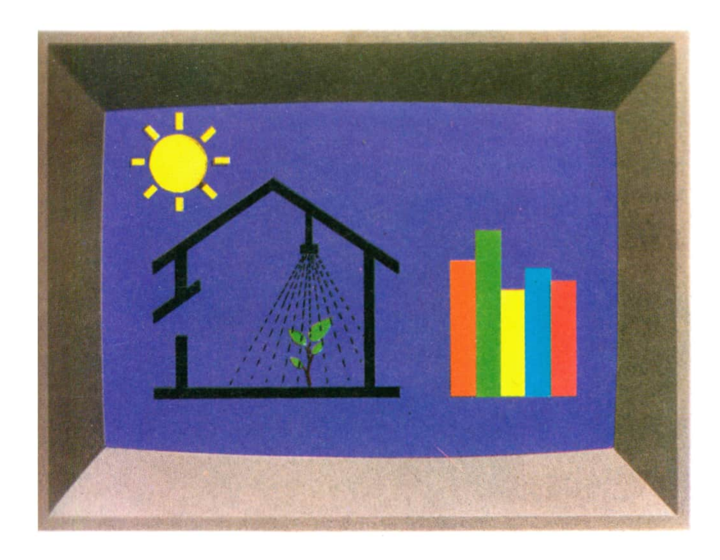
## Ciencias Naturales

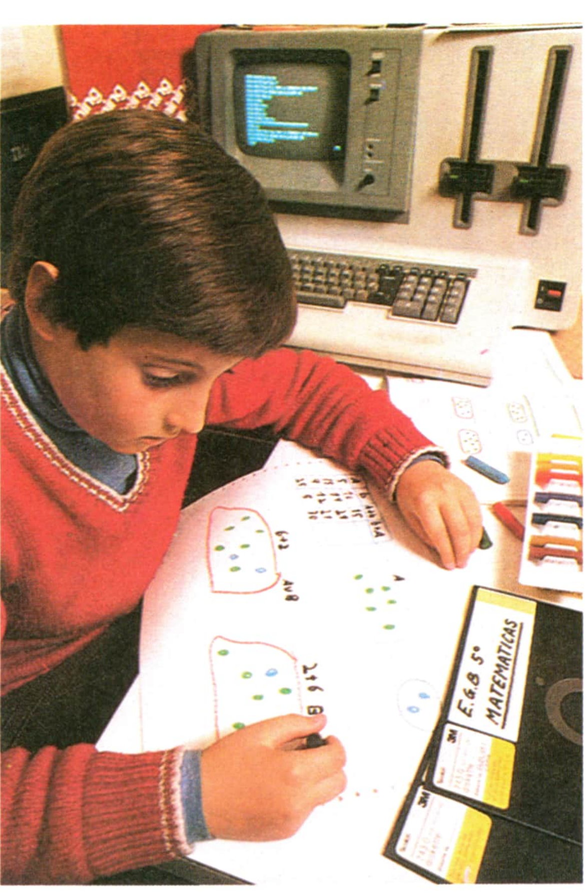
Vamos a enseñarle a la computadora los huesos de que está formado el cuerpo humano. Es un poco más complicado, porque en el ejemplo de las capitales y los países, un país sólo tiene una capital, mientras que un brazo tiene muchos huesos, una pierna tiene todavía más y la cabeza otro número diferente. Sin embargo, también puede hacerse.

```
10 REM SAVE "huesos.bas"
20 DATA cabeza, 13, tronco, 11, brazos,
6, piernas, 8:REM
(le decimos las partes en las que está
dividido el cuerpo y el número de
huesos que tiene)
30 DATA frontal, parietal, occipital,
temporal, etnoides, esfenoides,
lacrimales, nasales, pomulos, cornetes,
palatinos, vomer, maxilar: REM (le
decimos cuáles son)
40 DATA...:REM (lo mismo, pero con los
huesos del tronco)
50 DATA...:REM (lo mismo, pero con los
huesos de los brazos)
60 DATA...:REM (ídem con los huesos de
las piernas)
```

Es decir, que primero debemos darle todos los datos en forma de DATA. A partir de aquí, sólo tenemos que decir que nos pregunte lo que queremos saber:

```
100 PRINT "de que parte quieres saber
los huesos"
110 INPUT parte$
120 IF parte$ = "" THEN END
130 RESTORE 20:parte = 0
140 FOR i = 1 TO 4
150 READ par$, numero:REM
(que lea la parte y el número de huesos
que tiene)
160 IF par$ = parte$ THEN parte = i
170 NEXT i
180 IF parte = 0 THEN PRINT
"no existe esta parte": GOTO 100
190 IF parte = 1 THEN RESTORE 30:
REM
(como ya sabemos de qué parte se trata,
sólo tenemos que decirle a qué
instrucción tiene que ir a leer los
huesos)
200 IF parte = 2 THEN RESTORE
40:REM
210 IF parte = 3 THEN RESTORE 50;
REM
220 if parte = 4 THEN RESTORE 60:REM
225 PRINT "los huesos de"; parte$; "son:"
230 FOR i = 1 TO numero: REM
(sabemos que esta PARTE tiene
NUMERO huesos)
240 READ hueso$
250 PRINT hueso$
260 NEXT i
270 GOTO 100
```






*Abajo, a la izquierda, esquema de un invernadero en el marco de un programa para el aprendizaje de las Ciencias Naturales que permite al alumno controlar las condiciones ambientales. Arriba, programa de Matemáticas para el aprendizaje de sistemas de ecuaciones con dos incógnitas.*

## Matemáticas

Vamos a hacer un programa para resolver problemas de dos ecuaciones con dos incógnitas. Para ello, emplearemos el método llamado de «sumas y restas».
Empezaremos con un ejemplo:
Si multiplicamos la primera ecuacion por 3 y la segunda por 2, obtendremos:

$$\left.\begin{array}{l}2x+3y=22\\3x-2y=7\end{array}\right\}$$

$$\left.\begin{array}{l}6x+9y=66\\6x-4y=14\end{array}\right\}$$

Si restamos, nos quedará la ecuación:

$$0x+13y=52$$

de donde

$$y=\frac{52}{13}\ ;\ y=4$$

Es decir, se trata de multiplicar los coeficientes de una ecuación por el de las *x* de la otra y los de ésta por los de la *x* de la primera, cambiar el signo de todos los de una, sumarlos y dividir la suma del termino independiente por la suma del coeficiente de la *y*.

```
10 REM SAVE "sistema"
15 PRINT "primera ecuacion"
20 PRINT "entra el coeficiente de la x"
30 INPUT A1:REM
(entramos el coeficiente de la x)
35 IF A1=0 THEN END: REM
(señal de parada)
40 PRINT "entra el coeficiente de la y"
50 INPUT B1
55 IF B1=0 THEN PRINT "no vale":
GOTO 50: REM
(no es un sistema de ecuaciones)
60 PRINT "entra el termino
independiente"
70 INPUT C1: REM (entramos el término
independiente)
80 PRINT "segunda ecuacion"
90 PRINT "entra el coeficiente de la x"
100 INPUT A2
105 IF A2=0 THEN PRINT "no vale":
GOTO 100
110 PRINT "entra el coeficiente de la y"
120 INPUT B2
130 PRINT "entra el termino
independiente"
140 INPUT C2
150 A3=—A1*A2: REM (multiplicamos
la primera por A2 y le cambiamos el
signo)
160 B3=B1*A2: REM
170 C3=—C1*A2: REM
180 A4=A2*A1
190 B4=B2*A1: REM (lo mismo con la
segunda sin cambiar el signo)
200 C4=C2*A1:REM
210 A5=A3+A4
220 B5=B3+B4:REM (sumamos)
230 C5=C3+C4
235 IF B5=0 AND C5=0 THEN PRINT
"las dos ecuaciones son iguales": GOTO
15
236 IF B5=0 AND C5<>0 THEN
PRINT "no puede ser, no hay
soluciones": GOTO 15
```

*Anticiparse a los posibles errores*
Estas dos instrucciones se dan porque si B5=0 al dividir por B5 nos encontraríamos dividiendo por 0, que, como sabes, no se puede hacer. Si C5 fuera también 0, las dos ecuaciones son la misma; como por ejemplo:

x+y=2, 2x+2y=4

Si C5 no fuera cero, significaría que no hay solución posible, como por ejemplo:

x+y=2  x+y=3

ya que si la suma vale 2, no puede valer 3 al mismo tiempo.

```
240 Y=C5/B5:REM (dividimos)
250 X=(C1—B1*Y)/A1:REM
(despejamos la x)
260 PRINT "x=";X
270 PRINT "y=";Y
280 GOTO 15:REM (vuelta a empezar)
```

## Idiomas

Vamos a hacer un diccionario, de inglés por ejemplo, aunque igual podría ser de otro idioma. Para ello, vamos a necesitar un nuevo concepto: los vectores o *arrays*, como se llaman en inglés, que no son más



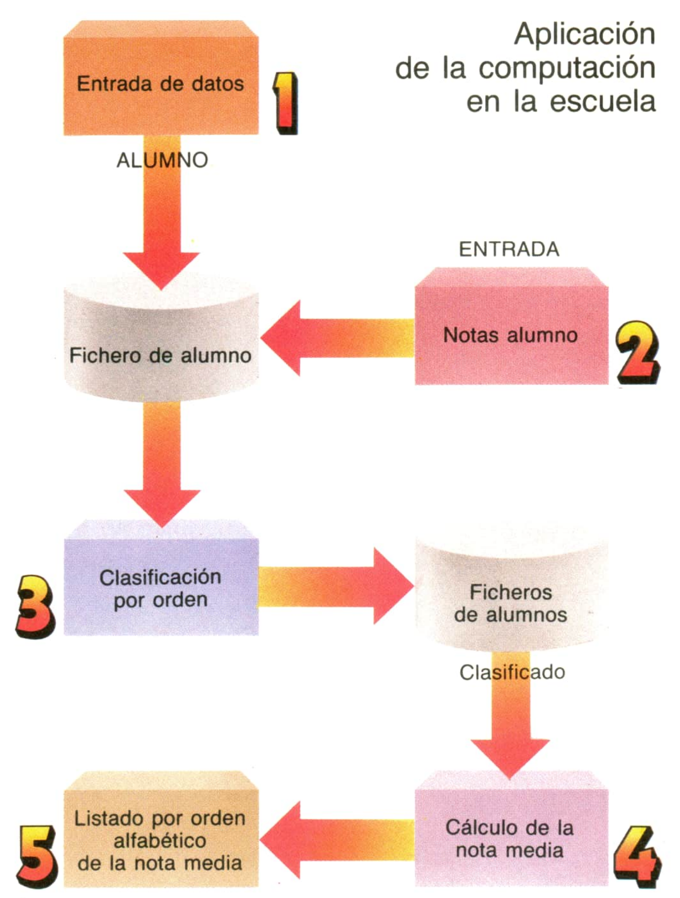
*Niños trabajando en el área de gramática con la ayuda del lenguaje LOGO. Este lenguaje ofrece, según los pedagogos, grandes posibilidades, dada su enorme capacidad interactiva.*

```
10 REM SAVE "diccio.bas"
15 DIM C$(1ØØØ),I$(1ØØØ):REM
(que reserve espacio para 1.000 palabras castellanas, C$(1ØØØ), y para 1.000 inglesas,
I$(1ØØØ))
17 NUM = Ø:REM (número de palabras que tiene nuestro vocabulario)
2Ø PRINT "palabra castellana": NUMERO = Ø
3Ø INPUT PC$:if pc$ = "" THEN END
4Ø FOR I = 1 TO NUM
5Ø IF C$ (I) = PC$ THEN NUMERO = I:I = NUM:REM
(al decirle que I vale NUM le estamos ordenando que pare el bucle)
6Ø NEXT I
7Ø IF NUMERO >Ø THEN 18Ø: REM (si está, que se vaya a enseñarla)
8Ø PRINT "no la tengo, quieres ponerla":REM         (pregunta obligada, pues nos
9Ø INPUT RESPUESTA$:REM                             podemos haber equivocado
1ØØ IF RESPUESTA$ < > "SI" THEN 2Ø: REM              al teclearla).
11Ø PRINT "palabra inglesa"
12Ø INPUT PI$
13Ø IF PI$ = "" THEN PRINT "no vale": GOTO 11Ø:REM   (debe tener traducción)
14Ø NUM = NUM + 1:REM                                (ya tenemos una palabra
                                                     más)
15Ø I$(NUM) = PI$:REM                                (guardamos la castellana en
                                                     su sitio)
16Ø C$(NUM) = PC$:REM                                (ídem con la inglesa)
17Ø GOTO 2Ø:REM                                      (que vaya a pedir otra
                                                     palabra)
18Ø PRINT "su traduccion es"; I$(NUMERO):REM         (que enseñe la traducción)
19Ø GOTO 2Ø
```




que un nuevo sistema de llamar a las variables.
Con las variables que conoces hasta ahora no podrías hacer un diccionario, porque cada palabra y su traducción tienen que ser variables diferentes y no podrías llamarlas palabra1$, palabra2$..., palabra2341$. Lo entiendes, ¿verdad?

*La solución DIM*
Pues bien, si le decimos al compilador que pondremos mil palabras seguidas, y que las llamaremos palabras$(1) a la primera, palabra$(2) a la segunda y palabra$(2341) a la palabra número 2341, podemos cambiar el número que hay entre paréntesis por una variable y escribir palabra$(I), que el compilador ya se espabilará. La instrucción para que reserve suficiente espacio de memoria se llama DIM y se escribe: 10 DIM palabra$(100), que quiere decir que reserve espacio para 100 variables alfabéticas. Para hacer el dicionario, tenemos que prever dos casos: uno, poner las palabras nuevas y su traducción, y dos, encontrar las palabras que ya hemos introducido y buscar la traducción.




*El dibujo muestra el proceso que ha de seguirse para ejecutar un programa de aplicación de informática a la escuela. 1) Entrada de los datos del alumno. 2) Entrada de las notas de cada alumno. 3) Clasificación del fichero por orden alfabético. 4) Cálculo de la nota media. 5) Listado por impresora.*

### *Archivos*

Este programa sólo tiene un inconveniente: cuando lo terminas, se pierden las palabras que has entrado. Como esto no podemos permitirlo, es el momento de que aprendas a guardar información permanente, de la misma manera que guardas los programas. Se consigue con los archivos.

### *¿Cómo funcionan?*

Se trata de guardar los datos que entras por programa. Para ello, sólo hacen falta cuatro instrucciones. Una para *abrir* el archivo, otra para *cerrarlo,* una tercera para *grabar* información y la última para *leerla.*
Al abrir un archivo le asignas un número, para que la máquina sepa a cuál te refieres; también le tienes que indicar si será un archivo del que leerás datos o si, al contrario, los vas a guardar.
Con todo ello, podemos hacer ya el diccionario permanente. Sólo tenemos que empezar el programa abriendo el archivo para lectura, leerlo y ponerlo en los vectores C$() y I$() y, al acabar de ejecutar el programa, abrirlo para grabación (para ello, habrá que cerrarlo primero), grabar los datos y luego cerrarlo otra vez.
Para evitar tocar el programa que ya tenemos, utilizaremos una rutina de las que hablábamos al principio. En Basic se llaman GOSUB y se escriben así:

```
Diccionario, segunda parte
Después de la instrucción 17 NUM=Ø
```

escribiremos:

```
18 GOSUB 1ØØØ:REM leer:REM
(que se vaya a la 1ØØØ y que cuando
acabe vuelva; es la llamada a una rutina
en Basic; vamos ahora a escribir la
rutina de leer de un archivo).
```

### *Abrir y cerrar un archivo*

```
1ØØØ REM leer el vocabulario
1Ø1Ø OPEN "i",1,"diccio.reg":
Es la instrucción para abrir un archivo;
```

Siempre hay que decir si es de lectura, I, o de escritura, O; luego se le indica un número de orden para que la máquina sepa a cuál corresponde y, por último, se pone el nombre del archivo

```
1Ø2Ø WHILE NOT EOF (1):
```

WHILE significa mientras, en inglés, y es el MIENTRAS que le enseñamos a Jane; el bloque de instrucciones que debe ejecutar se acabará cuando encuentre la instrucción WEND
EOF(1) significa END OF FILE, es decir, final de archivo; el significado de esta instrucción es "Mientras no encuentres el final de archivo".



*La introducción de los microordenadores en el trabajo escolar es un importante desafío para la renovación pedagógica, ya que la interactividad de las computadoras estimula la capacidad creativa y de investigación del alumno, fomentándole una actitud crítica frente al mundo actual.*

```
1Ø3Ø num = num + 1:REM
(suma uno al número de palabras que
tienes)
1Ø4Ø INPUT# 1, c$(num):REM
(lee y PON en las castellanas)
1Ø48 INPUT# 1,i$(num):REM
(lee y PON en las inglesas)
1Ø6Ø WEND:REM
(fin de las acciones del WHILE)
1Ø7Ø RETURN:REM
(fin de la rutina)
```

Como puedes ver, se lee del disco igual que de la pantalla; únicamente se añade el signo «#» y el número del archivo.

*Grabar y cerrar un archivo*
Para grabar este archivo al acabar, hay que cambiar la instrucción 30, que era:

```
3Ø INPUT PC$:IF PC$ = "" THEN END
```

En su lugar, debes poner

```
3Ø INPUT PC$:IF PC$ = ""
THEN 2ØØØ:REM
(grabar y cerrar)
```

Las instrucciones de grabar y cerrar serán:

```
2ØØØ REM (grabar y cerrar)
2Ø1Ø CLOSE 1:REM cierra el 1
2Ø2Ø OPEN "o",1,"diccio.reg":REM
(abre el archivo para grabación)
2Ø3Ø FOR i = 1 TO NUM:REM
(todas las palabras que tienes)
2Ø4Ø PRINT# 1,c$(i):REM
(graba la palabra castellana)
2Ø5Ø PRINT# 1,i$(i):REM
(graba la palabra inglesa)
2Ø6Ø NEXT i
2Ø7Ø CLOSE 1:REM
(cierra el archivo 1)
2Ø8Ø END:REM PARA
```

## ORDENADOR VIENE DE ORDENAR

El programa anterior sigue teniendo fallos. No existe el programa perfecto. El defecto en este caso es que las palabras quedan almacenadas en el orden en que las has entrado. Pero si quieres ordenarlas por orden alfabético, es bastante fácil. Hay muchas maneras de ordenar. Vamos a ver cuál es la más sencilla, lo que no significa que sea la más rápida. Se llama el «método directo».




*Arriba, clase de matemáticas realizada con ayuda del ordenador. Se ha insertado, además, la pantalla con la puntuación otorgada a tres alumnos e instrucciones para la corrección de los errores y continuación del programa.*

**Método directo**

Imagina que estás jugando a cartas. ¿Cómo las ordenas? Miras la primera, luego la segunda. Si ésta es más pequeña que la anterior, simplemente las cambias de sitio y luego continúas con la siguiente. Cuando has comparado la primera con todas las demás, estás seguro de que la primera es la más pequeña. Luego repites la operación con la segunda, la tercera, etc., hasta llegar a la penúltima, pues la última no hace falta que la compares.

```
1Ø REM SAVE "ordenar.bas"
2Ø GOSUB 1ØØØ: REM LEER
(ya sabes cómo se hace)
3Ø FOR i = 1 TO NUM-1:REM
(desde la primera hasta la última)
4Ø FOR i1 = i1 + 1 TO NUM:REM
(desde la siguiente de la que comparas
hasta la última)
5Ø IF pc$(i) > pc$(i1) THEN GOSUB
11ØØ:REM
(si la que tiene que ser más pequeña es
mayor que la que tiene que ser mayor,
debes cambiarla; luego lo haremos)
```

```
6Ø NEXT i1
7Ø NEXT i
8Ø GOTO 2ØØØ:REM (grabar y acabar)
```

*Cambio de sitio de las variables*

```
11ØØ REM (cambio de variables)
111Ø a$ = pc$(i):REM
(guardamos el valor de pc$(i) en otra
variable)
112Ø pc$(i) = pc$(i1):REM
(ponemos la segunda en la primera)
113Ø pc$(i1) = a$:REM
(ponemos en la segunda la primera)
114Ø RETURN:REM (vuelta)
```

*Listar o imprimir el diccionario*
Si posees impresora y quieres tener tu vocabulario escrito en un papel, no hay nada más fácil. Intercala la instrucción:

```
75 GOSUB 3000:REM LISTAR
```

La rutina de listar es así:

```
3000 REM LISTAR
3010 FOR i=1 TO NUM
3020 lPRINT C$(i);TAB(40);i$(i):REM
(lPRINT es la orden para que escriba
por la impresora y TAB(40) significa que
lo haga a partir de la columna 40)
3030 NEXT i
3050 RETURN
```

*El programa "Deberes.bass"*
Para pasar de un programa a otro, debes hacerte un menú como hiciste en los cálculos de geometría. Pero para evitar tener un programa muy grande, puedes hacer el programa DEBERES:

```
10 REM SAVE "deberes.bas"
20 cls:REM (esta instrucción
borra la pantalla)
30 PRINT
"*********** DEBERES ***********"

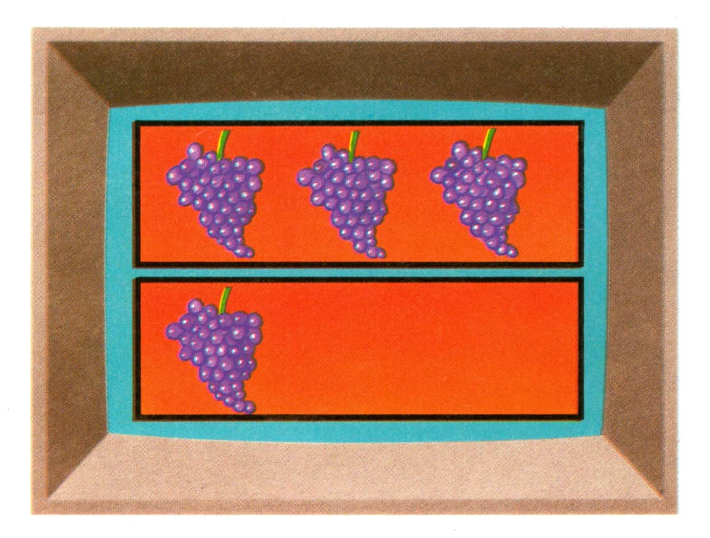
40 PRINT "GEOMETRIA...1"
50 PRINT "MATEMATICAS...2"
60 PRINT "GEOGRAFIA...3"
70 PRINT "CIENCIAS...4"
80 PRINT "IDIOMAS...5"
90 PRINT:REM
   (que deje una línea en blanco)
100 PRINT "SELECCIONA (0=FIN)"
110 INPUT RES
120 IF RES=0 THEN END
130 IF RES<0 OR RES>5 THEN PRINT
"respuesta incorrecta"
GOTO 100
120 ON RES GOTO 130, 140, 150, 160,
170
170 RUN "Calculo":REM
(ejecutará el programa «Cálculo»)
180 RUN "Sistema"
190 RUN "Capital"
200 RUN "Huesos"
210 RUN "Diccio."
```

De esta manera, del programa Deberes pasas al que hayas seleccionado. Claro que para volver de éste al de Deberes debes cambiar todas las instrucciones "END" por —RUN "deberes". Al llegar a esta instrucción, volverás al menú.

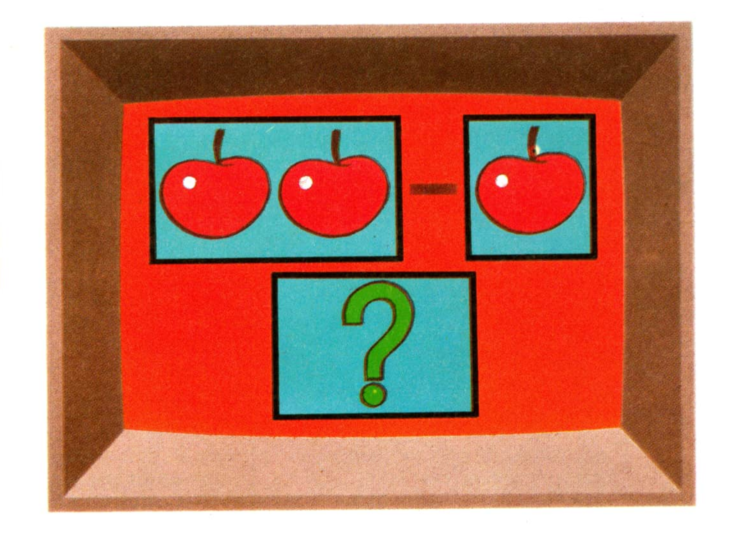
## Las cuentas de mamá

Siempre que tengas que hacer un programa nuevo, piensa si se parece a algo que ya hayas hecho antes. ¿A qué se parece llevar las cuentas? Aparentemente a nada relacionado con los deberes; sin embargo, si lo piensas bien, comprobarás que sí existe parecido. Llevar las cuentas es contestar a una serie de preguntas:

— Cuánto te has gastado en COMIDA.
— Cuánto te has gastado en ZAPATOS.
— Cuánto te has gastado en ROPA.



*Los dibujos de las pantallas superior e inferior están orientados al desarrollo del cálculo para niños muy pequeños. El ordenador ofrece grandes posibilidades para aplicar este tipo de juegos matemáticos en los distintos niveles de la enseñanza. Por ejemplo, un programa para llevar las cuentas de mamá como el que aparece en las páginas 70 y 71.*





*Dentro de las numerosas posibilidades que puede ofrecerte un microordenador, se halla la de la realización de gráficos mediante programas de diseño asistido, que, no obstante, parecer muy complejos algunos pueden realizarse con BASIC. Los microordenadores actuales disponen de un sistema operativo para ejecutar programas de software de diseño de gran impacto gráfico.*

Y todas las que se te ocurran. Es similar al programa de las capitales, sólo que ahora la máquina te hará las preguntas y tú debes darle las respuestas. Lo haremos del siguiente modo:

```
10 REM SAVE "cuentas.bas"
20 DATA comida, zapatos, ropa
22 RESTORE 20
23 FOR i=1 to 3
24 READ co$:PRINT "cuanto has
gastado en";co$
25 INPUT ga
26 NEXT i
```

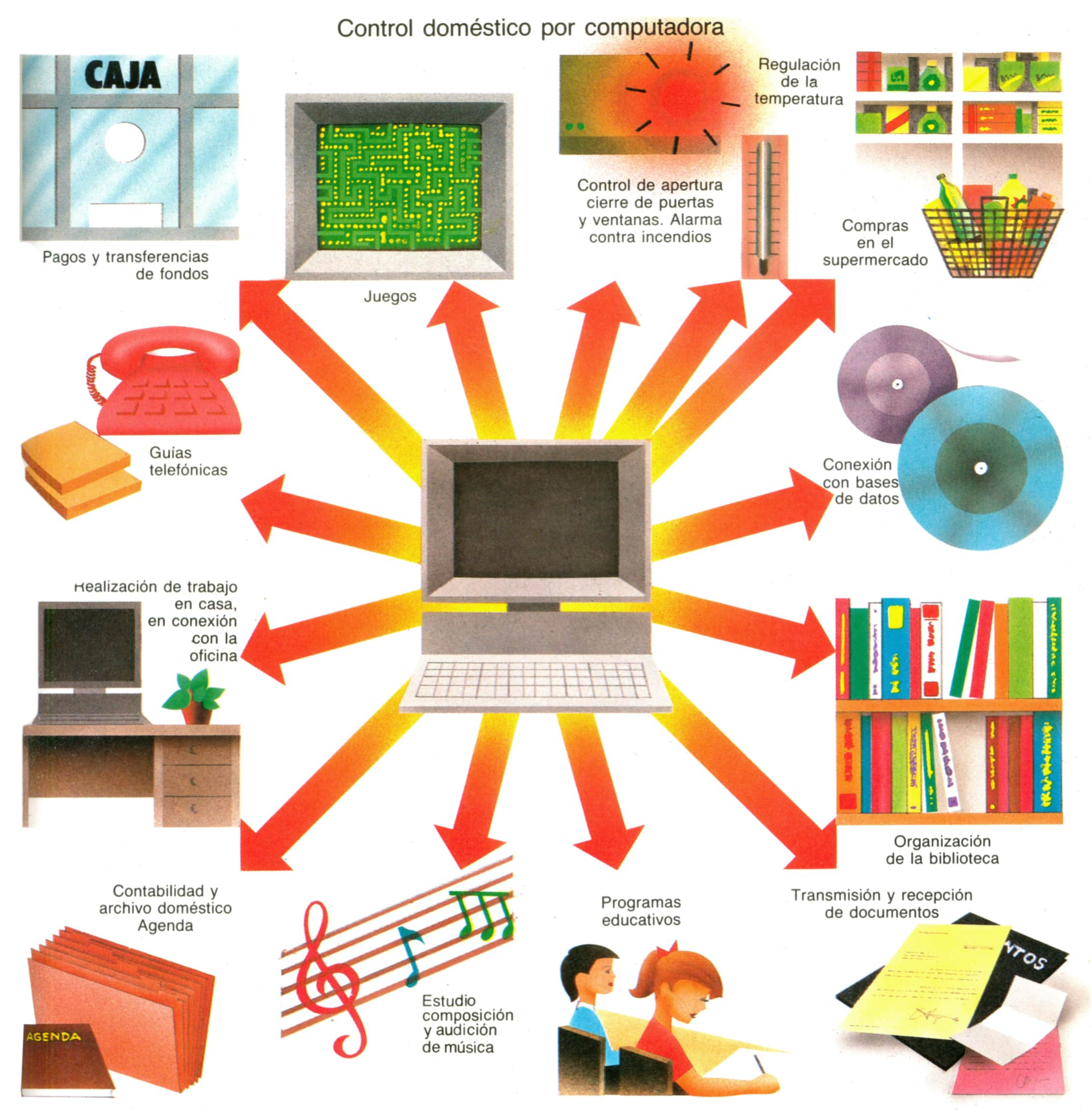
*Suma los gastos*
Como puedes ver, ya tienes la lista de preguntas y respuestas. Ahora sólo te falta sumar los gastos y ya tienes el gasto del día. Pero lo interesante es guardar las sumas de todo el mes. Para ello, tendrás que utilizar dos vectores, como en el diccionario. En uno guardarás el concepto y en otro, los gastos mensuales.

```
15 DIM co$(100),DI$(100)
20 DATA, comida, zapatos, ropa
22 RESTORE 20
23 FOR i=1 to 3
24 READ co$(i):PRINT co$(i):PRINT
"cuanto has gastado en";cos$(i)
25 INPUT ga#:diario#=diario#+ga#
:di#(i)=di#(i)+ga#:REM (de esta
manera, sumas el de hoy y los
anteriores)
26 NEXT i
```

*Guarda los gastos*
Para poder guardar los gastos de un día para otro, tendrás que ir al disco, es decir, abrir un fichero. Como ya sabes, porque has aprendido a hacerlo, el programa quedará así:

```
10 REM SAVE "cuentas.bas"
20 DIM co$(100),di#(100)
30 DATA comida, ropa, zapatos: REM (es
mucho mejor que le preguntes a tu
mamá cuáles son los gastos que tiene)
40 RESTORE 20:numero=3:REM
(es conveniente poner en variables todos
los números fijos que utilizas)
50 GOSUB 1000:REM
(abrir fichero, que ya sabes cómo se
hace; es más, lo tienes sólo con que
cambies el nombre del fichero y las
variables de lectura)
60 RESTORE 20
70 FOR i=1 TO número
80 READ co$(i):PRINT "cuanto has
gastado en";co$(i)
90 INPUT ga#:diario#=diario#+ga#
:di#(i)=di#(i)+ga#
100 NEXT i
110 GOSUB 2000:REM
(grabar y acabar)
```

*Escribe los gastos*
Puedes listarlo, como en el diccionario; sólo que esta vez tendrás que ir sumando




las cantidades que escribes. Y sabes, sin embargo, que es muy fácil:

```
1111 total# = total# + di#(i)
```

Y al final, ordenas:

```
1222 lPRINT "total";total#
```

*Si has seguido y aplicado a tu microordenador el programa para llevar los gastos de mamá, observarás que no es un programa difícil. Naturalmente, puedes complicarlo mucho más. Pero para esto ya están los programas que pueden comprarse a la casa que fabrica y vende los ordenadores. El dibujo muestra algunas de las funciones que en un futuro próximo podrá desarrollar el ordenador en casa, valiéndose simplemente de un sistema experto de control doméstico. Se trataría de complicar más y más nuestro programa, pero fíjate qué resultados. ¿Parece increíble, verdad?*

*Ampliación de la lista de gastos*
Si tu mamá ha olvidado un gasto, lo cual es bastante probable, tienes que cambiar el programa añadiéndole el dato que faltaba en la instrucción DATA y cambiar el valor de la variable NUMERO. Fíjate que así sólo lo tienes que hacer en una instrucción y no cada vez que aparece, como habría pasado si no hubieras ordenado que "NUMERO = 3"

*Borra los gastos de un mes*
Para terminar este programa, sólo te falta que tu mamá pueda borrar los gastos cada vez que cambia el mes. Es la rutina "poner a cero". Justo después de leer y antes de empezar a preguntar escribes

```
55 GOSUB 4000:REM (poner a cero)
4000 REM (rutina de poner a cero)
4010 FOR i = 1 to
numero:di#(i) = 0:NEXT i:REM
(dices que todos valen cero)
4040 RETURN
```

## DIBUJAR

Para que tu computadora aprenda a dibujar, debes tener una pantalla gráfica. Y lo mejor es que programes en el lenguaje LOGO, que parece hecho a propópsito para ello.
El lenguaje LOGO es interactivo, como el Basic, es decir, es un programa que entiende lo que le dices y obedece. El Logo de MS-DOS entiende el castellano y tiene un manual muy sencillo.
Como en el Basic, le llamas por su nombre; Logo, y el programa se empieza a ejecutar. La señal de que está listo es que aparece el signo: «?».
Le dices "DIBUJA", se borra la pantalla y aparece en medio una tortuga, una tortuga obediente y bastante lista. Obedece todas las órdenes que se te ocurran para que dibuje: que avance, que retroceda, que borre, que gire a la derecha o a la izquierda. Y todo esto lo puede hacer dibujando o no, es decir, como si le dijeras "LEVANTA LAPIZ", "BAJA LAPIZ".
Las órdenes se las das en castellano: "AVANZA 30". El 30 representa las posiciones que debe avanzar. Para girar, se utilizan los grados normales.
La tortuga también tiene la orden REPITE /n/ [/acción/].
Es decir que si le dices REPITE 360 [AVANZA 1 GIRA DERECHA 1], la tortuga repite 360 avanza 1 unidad y gira un grado, o sea, dibuja un círculo.

## RUTINAS

La tortuga también recuerda lo que le dices. Para ello, debes decirle PARA CIRCULO y el interrogante se convierte en >, lo que quiere decir que está dispuesta a aprender. Le dices entonces:

```
REPITE 360 [AVANZA 1
GIRADERECHA 1]
FIN
```

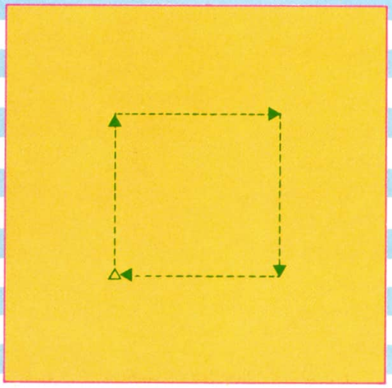
Las cosas que puedes hacer con Logos

Puedes darles tantas órdenes como quieras, pero debes acabar con la palabra FIN. Si le dices luego CIRCULO, se acuerda de lo que significa y te dibuja un círculo.
Si escribir GIRADERECHA te cansa, la tortuga también sabe que se lo puedes abreviar con GC y que todas las órdenes se pueden escribir con dos letras, generalmente las iniciales.
Puesto que la tortuga se acuerda de todo lo que le digas, para hacer cualquier dibujo, sólo tienes que dividirlo en sus figuras más sencillas. Por ejemplo, un televisor son dos cuadrados (uno dentro

### PROGRAMAS LOGO EN LOS QUE SE REALIZAN FIGURAS GEOMÉTRICAS CON LA TORTUGA

Para que la TORTUGA dibuje un cuadrado se podría construir el siguiente programa LOGO:

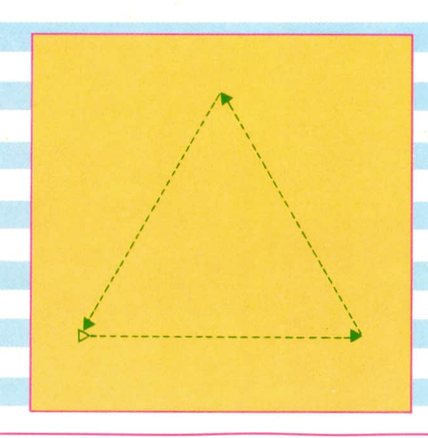
```
CS       borrado de la pantalla
FD 30    indica a la tortuga que desde donde está
         avance 30 unidades en la dirección que señale extremo
RT 90    indica a la tortuga que varíe su posición 90° a la derecha
FD 30    le indica que avance 30 unidades
TR 90    le indica que varíe su posición 90° a la derecha
FD 30    y así sucesivamente hasta formar el cuadrado tal como
         se indica en la figura
RT 90
FD 30
END
```



Veamos ahora cómo puede construirse un triángulo:

```
CS       borrado de pantalla
FD 30    avance de 30 unidades
LT 120   variación de 120° a la izquierda
FD 30    avance de 30 unidades
LT 120   variación de 120° a la izquierda
FD 30    avance de 30 unidades
LT 120   variación de 120° a la izquierda para quedar en su posición inicial
END
```

Con este programa podemos construir el triángulo de la figura.



del otro) y tres círculos pequeños en la parte baja de los dos cuadrados. Así que le dices PARA TELEVISOR y le explicas a continuación lo que tiene que hacer, es decir: levantar el lápiz, irse a una esquina, bajar el lápiz, dibujar un cuadrado, levantar el lápiz, girar 45 grados, irse ligeramente al centro, bajar el lápiz, dibujar un cuadrado, subir el lápiz, moverse hacia abajo, bajar el lápiz, y repetir tres veces: dibujar un círculo, levantar el lápiz, irse ligeramente a la derecha, bajar el lápiz.

## VÍDEOJUEGOS

También le podemos enseñar a hacer un vídeojuego. Uno sencillito, claro, para empezar. Te proponemos el ejemplo de las motos de la película "TRON". Por si no lo recuerdas, es un videojuego para dos jugadores, en el que cada uno conduce una moto que corre y deja una estela detrás de ella, como si fuera levantando una pared. El que choca con la pared pierde.
Veamos qué es lo que nos hace falta. Lo primero es saber cómo le diremos a la máquina que la moto ha chocado. Es fácil: un punto en la pantalla es una posición de memoria, y por lo tanto podemos preguntar si hay algo allí. Si lo hay, significa que la moto no puede pasar. Esta instrucción se llama POINT (/fila/,/columna/), es decir, las coordenadas del punto de la pantalla.

*Instrucciones gráficas*
La placa de gráficos entiende esta instrucción. También tenemos la instrucción LINE (/fila/,/columna/) — (/fila/,/columna/) que nos traza la línea definida por los dos puntos.
Tenemos así mismo el PSET (/fila/,/columna/) que pone un punto en este sitio. La pantalla tiene 500 x 200 puntos de longitud.
Sabiendo todo esto, ya podemos empezar. Primero tenemos que delimitar la pantalla. Hagamos, pues, esta rutina.

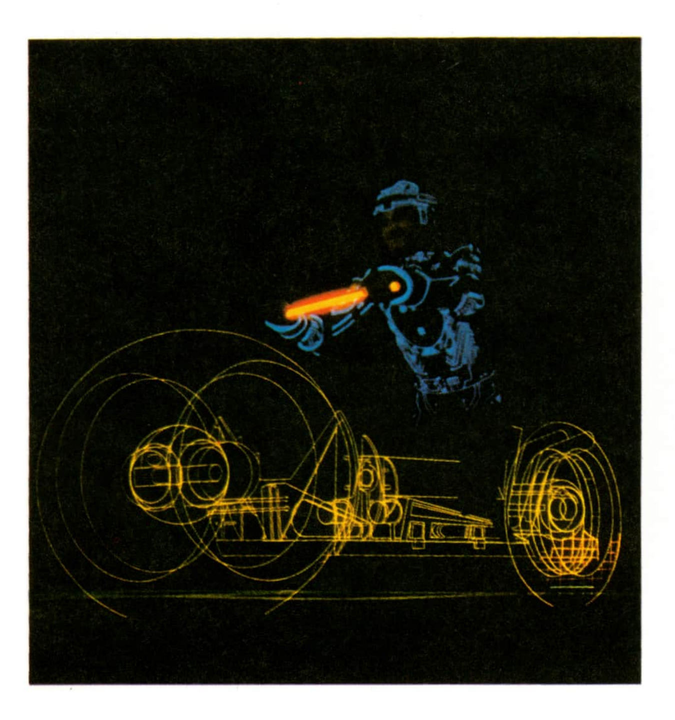
```
100 REM (rutina de delimitar la
pantalla; la pared exterior)
1010 CLS:REM (borrar la pantalla)
1020 LINE (10,10)—(550,199),,b:REM
(esta manera de escribir la orden LINE
ordena que nos haga un rectángulo)
1030 LOCATE 1:REM
(que sitúe el cursor en la primera línea)
1040 PRINT "llevo matados";se;"llevo
matados";pr
1050 return
```

Al empezar el juego, aparece la pantalla y las dos motos se ponen en marcha, la una contra la otra. Tendremos que teclear las rutinas de situarse en posición y no moverse.

```
150 REM (ponerse en posición)
1510 h = 400:v = 180:h2 = 100
:v2 = 80:IH = —1:IV = 0:IH2 = 1
:IV2 = 0:REM
(las coordenadas de donde queremos que
se sitúen las motos)
IH,IV,IH2,IV2 (son las variables que
indican en qué dirección se mueven las
motos. Ya verás como actúan).
```

*Movimientos*

```
1520 RETURN.
200 REM rutina de moverse la moto de
la derecha
205 H2 = H2 + IH2:V2 = V2 = IV2:REM el
avance en la dirección indicada
```



*Fotograma de la película «Tron». Es un vídeojuego para dos jugadores. en el que cada uno conduce una moto que corre y deja una estela detrás de ella, como si fuera levantando una pared. El que choca con la pared pierde.*

```
210 IF POINT(h2,v2) = 1 THEN
pr = pr + 1:y = 0:x = 0:GOSUB
100:RETURN:REM (el choque)
220 PSET (h2,v2)
230 RETURN
300 REM (rutina de moverse la moto de
la izquierda)
305 H = H + IH:V = V + IV:REM
(el avance según la dirección indicada)
310 IF POINT (h,v) = 1 THEN
se = se + 1:x = 0:y = 0:GOSUB 100:REM
(el choque)
320 PSET (h,v)
330 RETURN
```

*Mover las motos*

Ahora tenemos que disponer los mandos para mover las motos. El jugador de la derecha puede mover 8 = arriba, 2 = abajo, 6 = derecha, 4 = izquierda y el jugador de la izquierda W, Z, S, A para lo mismo. El sistema de entrada de datos será el mismo que empleamos para el piano.

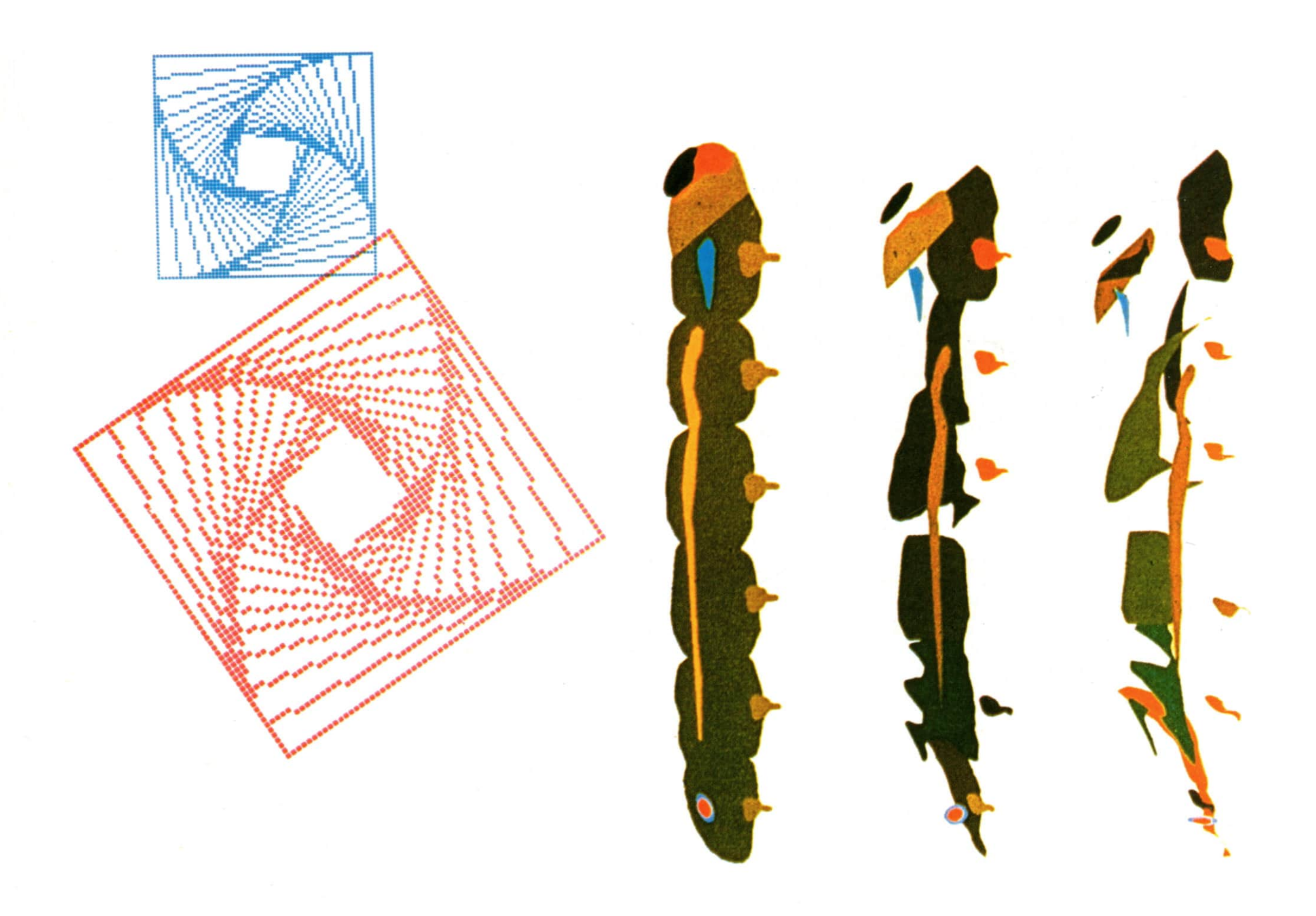
```
10 REM SAVE "tron.bas"
15 SCREEN 2:REM (la instrucción para
que la pantalla se ponga en gráficos)
20 a$ = "WZSA8264"
```

Después hay que detectar el movimiento:

```
500 REM (detectar el movimiento)
510 b$ = INKEY$
520 IF b$ = "" THEN RETURN:REM (si
nadie toca ningún mando, las dos motos
siguen su carrera)
530 a = INSTR (a$,b$):IF a = 0 THEN
RETURN:REM (la tecla tocada no es
válida)
540 ON A GOTO 450, 460, 470, 480,
550, 560, 570, 580,
550 REM (la moto de la derecha se
mueve hacia arriba)
552 IV = —1:IH = O:GOSUB 300:RETURN
:REM
(ve a moverte y luego vuelve)
560 REM (la moto de la derecha se
mueve hacia abajo)
562 IV = 1:IH = O:GOSUB 300:RETURN
570 REM (la moto de la derecha se
mueve a la derecha)
572 IHN = 1:IV = O:GOSUB 300:RETURN
580 REM (la moto de la dereoha se
mueve a la izquierda)
582 IH = -1:IV = O:GOSUB 300:RETURN
```

Los movimientos de la moto de la derecha son exactamente iguales siempre que cambies las iv por iv2 y las ih por ih2, y donde decías 'GOSUB 300' pongas 'GOSUB 200'. El programa queda entonces así:

```
10 REM SAVE "tron"
20 a$ = "WZSA8264":se = 0:pr = 0:REM
(ponemos a cero los contadores de
muertos)
30 GOSUB 1000:REM (la pantalla)
40 GOSUB 1500:REM (ponerse en
posición)
50 GOSUB 500:REM (detectar el
movimiento)
60 GOSUB 200:GOSUB 300:REM
(moverse)
70 GOTO 50
110 REM (el choque)
110 GOSUB 1000:REM (la pantalla)
120 GOSUB 1550:REM (ponerse en
movimiento)
130 RETURN
```



## GLOSARIO

**Ábaco** Máquina de calcular que ya utilizaban los antiguos caldeos.
**Acceso** Acto por el que se accede a la información almacenada en una computadora. Puede ser *secuencial,* cuando hay que recorrer todos los datos que preceden a la información para obtenerla, o *directo,* cuando se puede acceder a la información directamente.
**Acumulador** Registro donde se efectúan operaciones.
**Ada** Lenguaje de programación. Recibe su nombre en honor de Ada Byron, una de las pioneras de la programación.
**Alfabeto** Conjunto de los diferentes caracteres que pueden emplearse en las computadoras (letras, dígitos, signos de puntuación o símbolos).
**Algoritmo** Descomposición en pasos u operaciones elementales de cualquier problema para su resolución óptima.
**Aplicación** Conjunto de programas creados para una computadora con el fin de agilizar trabajos de contabilidad, gestión de almacenes o cálculos complejos.
**Archivo** En un sentido amplio, todo lo que está guardado en los dispositivos de memoria externos. En esta definición, también entran los programas. En un sentido más limitado, sólo son los almacenes de datos.
**Asignación** Atribución de valor a una zona de memoria que normalmente se denomina variable.
**Arrays** V. VECTOR.
**Banco de datos** Conjunto de la información sobre un tema, de la que puede disponer cualquier usuario que esté conectado al sistema al que pertenece la información.
**Base de datos** Ficheros de datos a los que pueden acceder una computadora o varias terminales que permiten al usuario actualizar, acceder o recuperar la información contenida en ellos.
**Basic** Lenguaje de programación. Siglas de *Beginer's Allpurpose Symbolic Instruction Code.* Al tratarse de un lenguaje interpretado, es tal vez el más utilizado en el aprendizaje.
**Biblioteca** Conjunto de programas o rutinas archivadas en un dispositivo de almacenamiento externo de memoria, al que se puede acceder desde un programa o desde un compilador.
**Biestable** Llamado también *flip-flop* o báscula, es un dispositivo que cambia de estado a cada impulso que recibe.
**Binario** Sistema de numeración que sólo tiene dos números, el 0 y el 1.



*Los ordenadores actuales ofrecen enormes posibilidades para la generación de gráficos, que, en general, utilizan los lenguajes de programación estandarizados y de fácil aprendizaje como el BASIC. Este precioso gráfico representa las sucesivas fases del desarrollo de una mariposa mediante diseño asistido por ordenador, lo que en informática recibe el nombre de CAD-CAM. A la izquierda, resultado de un programa que genera una serie de cuadrados mediante el llamado «efecto de torsión de la imagen».*

**Bit** La más pequeña unidad de información. Sólo puede tener dos posiciones.
**Blanco** Carácter que representa el lugar que queda vacío entre dos palos.
**Bucle** Ejecución repetida de un determinado número de instrucciones de una subrutina, que se interrumpe cuando llega a una conclusión que anula la reiteración y provoca que se realice la instrucción que sigue al bucle.
**Bus** Circuito del ordenador por el que circula la información en forma de bits.
**Byte** También llamado octeto, porque es la reunión de 8 bits. Tiene la capacidad de almacenar una letra o un número.
**Cabezal** Dispositivo electrónico que sirve para leer o grabar información de o en una unidad de almacenamiento externo (diskette, cinta, etc.).
**Canal** Vía de comunicación entre la C.P.U. (unidad central de proceso) y los dispositivos de control de entrada/salida (impresora, diskette, hard-disk, etc.).
**Carácter** Cualquier símbolo utilizado para representar letras, cifras, signos de puntuación u otros signos especiales.
**Cargar** Introducir cualquier tipo de información en la memoria de la computadora.
**Cassette** Cartucho de cinta magnética que sirve para almacenar información.
**Cinta magnética** Tira de poliéster revestida de una emulsión magnética en la que se puede grabar información.
**Cobol** Lenguaje de programación de utilización exclusiva para programas de gestión empresarial.
**Comando** Orden o instrucción para que el procesador ejecute una operación específica.
**Compilador** Programa que traduce un lenguaje de programación al lenguaje máquina.
**Computación** V. INFORMÁTICA.
**Computadora** V. ORDENADOR.
**Contador** Dispositivo que permite contar el número de impulsos que recibe.

**Convertidor** Dispositivo electrónico que permite la conexión de unidades.

**CPU** *(Central Processing Unit).* Unidad central de procesos, que gestiona todas las operaciones y procesos de una computadora.

**Chip** Circuito integrado compuesto por un número cada vez mayor de transistores.

**Decodificador** Circuito electrónico que descifra los datos de entrada para que puedan ser utilizados por la computadora.

**Digitalizador** Dispositivo de entrada de datos que permite situar las coordenadas de un punto dentro de la memoria de un ordenador por la posición de un punzón conectado con la computadora.

**Dirección** Posición de una información determinada en la memoria, es decir, el valor numérico o alfabético con el que se puede identificar un dato contenido en aquélla.

**Disco duro** En inglés *hard disk,* disco magnético de un material rígido que permite almacenar una gran cantidad de programas o información.

**Diskette** Disco magnético flexible de pequeñas dimensiones, usado para almacenar información. También se llama, *floppy disk.*

**Display** Pantalla donde se visualiza la información.

**Doble precisión** En Basic, variables numéricas que permiten una precisión de 16 dígitos y ocupan 8 bytes de memoria.

**Editar** Componer o alterar un texto a través de la pantalla del ordenador, siguiendo un programa denominado editor.

**Editor** Programa que permite la realización de textos o programas mediante la pantalla del ordenador.

**Ensamblador** Programa que traduce programas escritos en lenguaje de bajo nivel a lenguaje máquina.

**Entero** Variable numérica que sólo permite almacenar números enteros.

**Entrada** Introducción de datos en la computadora mediante el teclado u otro periférico diseñado para ello (ratón, lápiz óptico, etc.).

**Etiqueta** Señal que se pone a una línea de un programa para poder referirse a ella.

**Fichero** Conjunto de datos o instrucciones almacenados en el diskette, la cinta u otro soporte magnético.

**Floppie** V. DISKETTE.

**Fortran** Primer lenguaje de programación de alto nivel. Empleado en investigación por su rapidez de ejecución.

**Grabar** Crear o producir una información mediante la máquina.

**Hardware** Conjunto de las partes de un ordenador que se pueden tocar.

**Impresora** Dispositivo de salida de datos escritos.

**Indicador** Palabra que designa las variables para que el compilador gestione las direcciones de memoria.

**Informática** Disciplina científica que estudia el manejo y tratamiento automático de la información mediante el uso de ordenadores.

**Insertar** Introducir una palabra, número, carácter o símbolo dentro de una frase o en un conjunto de elementos.

**Instrucción** Orden que ejecuta la C.P.U. (unidad central de proceso).

**Interface** Dispositivo que permite intercambiar información entre dos unidades de la computadora.

**Intérprete** Programa que traduce un lenguaje de alto nivel instrucción a instrucción y las va ejecutando sucesivamente.

**Lápiz óptico** Periférico de entrada de datos que consiste en un lápiz provisto de una célula fotoeléctrica, que permite a la computadora reconocer los caracteres cuando el lápiz los enfoca.

**Lector** Periférico que recoge información del papel o la cinta magnética y la introduce en el ordenador para su tratamiento.

**Lectura** Operación por la que un dispositivo obtiene información de una memoria.

**Lenguaje de alto nivel** Lenguaje de programción simbólico, parecido a los idiomas naturales, independiente del lenguaje máquina.

**Lenguaje de bajo nivel** Lenguaje de programación, próximo al lenguaje máquina, en el que cada instrucción corresponde a otra diseñada en lenguaje máquina.

**Lenguaje de programación** Sistema de signos y símbolos que, mediante un conjunto de reglas, permite la construcción de programas con los que la computadora puede operar.

**Lenguaje máquina** Lenguaje de programación de más bajo nivel con el que puede operar la computadora. Corresponde al sistema *binario.*

**Línea** Unidad con la que se mide un programa y que puede constar de una a más instrucciones.

**Listado** Conjunto de las informaciones suministradas en papel por la impresora del ordenador.

**Logo** Lenguaje de programación especialmente indicado para aprender a programar.

**Llamada** Instrucción de un programa que requiere la intervención de un subprograma perteneciente a otra zona de memoria de la que estaba trabajando.

**Llave** Carácter o grupo de ellos que sirven para identificar cada uno de los registros lógicos de un fichero.

**Macroinstrucción** Instrucción en lenguaje simbólico, que, después de pasar por el compilador, se transforma en lenguaje máquina.

**Máquina analítica** Máquina que Babbage diseñó y que, aunque no funcionó nunca, sentó las bases para el posterior desarrollo de la informática.

**MARK-1** Primera máquina que merece el nombre de ordenador.

**Matriz** Cuadrado de dos dimensiones en el que las entradas horizontales son las líneas y las verticales las columnas.

**Memoria** Dispositivo que permite almacenar temporal o permanentemente la información.

**Memoria central** Memoria principal de una computadora.

**Memoria de acceso aleatorio o directo** Tipo de memoria que se puede leer, grabar o borrar y modificar, y en la que el acceso a una posición no depende de la información anterior o posterior requerida. También se llama memoria RAM.

**Memoria de acceso secuencial** Tipo de memoria de lectura/escritura en la que, para encontrar un dato, hay que pasar por toda la información registrada que le precede (cassettes, diskettes, etc.).

**Memoria de sólo lectura** Tipo de memoria permanente, que viene dada con el ordenador, por lo que no se puede modificar. Sólo admite leer. Recibe también el nombre de *ROM.*

**Menú** Programa que sirve de enlace entre otros programas.

**Microordenador** Pequeña computadora cuya unidad central de proceso (CPU) es un microprocesador, por lo que tiene una capacidad muy limitada.

**Microprocesador** Procesador en una única pieza.

**MS-DOS** Sistema operativo utilizado en la mayoría de ordenadores PC compatibles.

**Multiproceso** Ejecución simultánea de varios programas a cargo de varias unidades centrales o procesadores.

**Multiusuario** Computadora que trabaja con más de un usuario a través de varios puestos de trabajo conectados a ella mediante una terminal provista de pantalla y teclado autónomos.

**Octeto** V. BYTE.

**Operador** Persona encargada de realizar las operaciones manuales que requiere una máquina. // Carácter que indica una operación aritmética o lógica, como los signos de sumar, restar, multiplicar, dividir, etc. (+, —, ×, :).

**Operando** Cada uno de los elementos que en una instrucción acompañan al código de operación.

**Ordenador** Máquina que trata automáticamente la información. Está compuesta de una unidad central de proceso (CPU), memorias y periféricos de entrada/salida de datos. Los hay de varios tipos: microordenadores, personales, domésticos, etc.

**Ordinograma** Representación gráfica mediante figuras geométricas (rombo, rectángulo, cuadrado, etc.) de un problema determinado para facilitar su solución.

**Palabra** Conjunto de bits que, como unidad elemental, puede manipular un ordenador (8, 16, 32, etc.).

**Pantalla** Dispositivo de salida de datos en el que se visualizan las informaciones que el usuario teclea o que genera el mismo ordenador.

**Parámetro** Variable que puede tomar un valor diferente cada vez que se ejecuta una subrutina en la que se utiliza dicha variable.

**Pascal** Lenguaje de programación utilizado en investigación y para aprender programación estructurada. Recibe este nombre en honor de Blaise Pascal, inventor de la primera máquina de calcular mecánica.

**Pascalina** Primer aparato para sumar y restar, inventado por Pascal.

**Periférico** Aparato, dispositivo o unidad que no forma parte de la CPU, pero que, conectado a ésta, permite almacenar información o dar entrada y salida a las informaciones que busca el usuario (pantalla, disco, impresora, etc.).

**Pista** Cada una de las bandas capaces de recibir información en un diskette, disco o cassette.

**Placa** Soporte donde se montan los circuitos electrónicos, es decir, los chips o los microprocesadores.

**Plotters** Dispositivo de salida de datos que sabe dibujar.

**Posición de memoria** Cada una de las unidades elementales que puede contener información en la memoria central, y que se define mediante direcciones de memoria.

**Procedimiento** Serie de pasos que deben seguirse para resolver un problema.

**Procesador** Conjunto de circuitos eléctricos que interpreta y ejecuta una sucesión de órdenes.

**Procesamiento** V. TRATAMIENTO.

**Programa** Conjunto de instrucciones que siguen un orden secuencial, mediante las cuales la computadora genera unas informaciones o resultados que interesan al usuario.

**Programación** Técnica de confección de programas.

**Programador** Técnico en programación y que confecciona programas de ordenador.

**Protocolo** Conjunto de reglas y métodos que hay que seguir para intercambiar información entre dos computadoras.

**RAM** *(Random Acces Memory)*. Parte de la memoria en la que se puede leer y escribir.

**Ratón** Dispositivo de entrada que permite moverse por la pantalla accionando una bola por encima de la mesa.

**Real** Variable numérica que permite el uso de decimales.

**Récord** V. REGISTRO.

**Registro** Unidad lógica de información.

**Relé** Dispositivo electromecánico que permite mover una palanca al darle corriente.

**Repertorio de instrucciones** Conjunto de instrucciones que forman parte de un lenguaje de programación.

**Rom** ***(Read Only Memory)*** Parte de la memoria que sólo se puede leer y que no se borra nunca.

**Rótulo** Carácter o conjunto de caracteres que se utilizan para identificar una sentencia, un dato o conjunto de ellos.

**Rutina** Parte de un programa que tienen una función completa.

**Salida** Cualquier información obtenida de la computadora a través de un periférico de salida de datos.

**Salto** Instrucción propia de ciertos lenguajes que interrumpe la secuencia lógica del programa y ayuda a saltar varias instrucciones para acortar el tiempo de programa.

**Sector** Cada una de las zonas en que se dividen las pistas de un disco.

**Segmento** Parte de un programa que se carga de una sola vez en memoria.

**Sentencia** V. INSTRUCCIÓN.

**Símbolo** Carácter o conjunto de ellos que sirve para representar una cantidad, operación, información, etcétera.

**Simulación** Representación gráfica de un proceso en la pantalla de la computadora.

**Sistema operativo** Programa o conjunto de programas que controla todas las operaciones que puede realizar una computadora.

**Software** La parte de un ordenador que no se puede tocar, es decir, los programas.

**Soporte** Material destinado a recibir información y mantenerla de forma que pueda ser leída por la computadora, como cintas, cassettes, diskettes, etcétera.

**Subrutina** Conjunto de instrucciones que se utiliza entre varios puntos de un programa o que puede insertarse en diferentes programas. También se conoce por *subprograma.*

**Supervisor** Programa que forma parte del sistema operativo de una computadora.

**Tabular** Desplazar el cursor a una posición determinada, saltando varias posiciones de pantalla, para visualizar la información a partir de esa posición.

**Tarjeta perforada** Soporte, normalmente de cartón, que sirve como entrada de datos a una computadora. Ya no se utiliza en la actualidad.
**Tecla** Cada uno de los elementos de que se compone el teclado de un ordenador.
**Teclado** Periférico formado por un conjunto de teclas utilizadas para introducir información en la computadora.
**Terminal** Sistema compuesto por un teclado, una pantalla y un circuito de control, que se conectan a una computadora y se utilizan para introducir y extraer datos de aquélla.
**Traductor** Programa compilador o ensamblador.
**Transferencia** Movimiento de datos de una unidad a otra o de una a otra posición de memoria.
**Transistor** Dispositivo electrónico que permite el paso o la ampliación de una señal eléctrica.
**Transmisión** Acto por el que se envían datos de un sistema y se reciben por otro, como el envío de datos de la computadora a la impresora.
**Tratamiento** Ejecución automática de una serie de operaciones sobre una información proporcionada a la computadora, que la devuelve procesada.
**Tratamiento de textos** Programa que permite entrar un texto en la computadora, y al que, posteriormente, se puede manipular, modificar, suprimir o insertar, signos, palabras, frases o párrafos enteros.
**Turbopascal** Lenguaje de programación derivado del Pascal.
**UAL *(Unidad Aritmético-Lógica)*** Parte de la CPU que controla las operaciones aritméticas y las decisiones lógicas.
**Unidad** Conjunto mecánico electrónico que sirve de periférico a una computadora y que, mediante la introducción en su interior de un soporte magnético (diskette, cinta, disco, etc.), puede almacenar o suministrar información a la computadora.
**Unidad central** Dispositivo que forma la parte más importante de un equipo informático. Está compuesto de unidad de control, unidad lógica y memoria, y puede dirigir un conjunto de periféricos.
**Unidad Central de Proceso.** V. C.P.U.
**Usuario** Persona que utiliza la computadora o se sirve de la información por ella suministrada.
**Válvulas de vacío** Primer dispositivo electrónico que permitió realizar la conexión de circuitos sin partes móviles.
**Variable** Dato de un proceso que puede tomar distintos valores durante la ejecución de un programa.
**Variable alfabética** Variable que no permite operaciones aritméticas.
**Variable numérica** Variable que permite las operaciones aritméticas.
**Vector** Sistema de designación de un conjunto de variables de las mismas características.
**Ventana** Cada una de las partes en que puede dividirse la pantalla de una computadora, y en la que puede visualizarse información relativa a diferentes tareas realizadas por la computadora.
**Verificación** Control de la validez de los datos suministrados por una computadora.
**Vía** V. Bus.
**Vídeo** En informática, pantalla de una terminal o de una computadora.
**Visualizar** Hacer aparecer información en la pantalla de la computadora.
**Zona** Parte de la memoria central que puede reservarse para una función determinada.

# ÍNDICE DE MATERIAS

# ÍNDICE ALFABÉTICO



